Two-dimensional computer simulation of in-situ combustion oil recovery process and iterative methods

Bahadir, Ahmet Refik

Award date:
1994

Awarding institution:
University of Bath

TWO-DIMENSIONAL COMPUTER SIMULATION OF IN-SITU COMBUSTION OIL RECOVERY PROCESS AND ITERATIVE METHODS

Submitted by

Ahmet Refik Bahadır

for the degree of PhD

of the

University of Bath

1994

A.R. Bahadır

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

To my wife Nisa and our daughter İrem

Abstract

In-situ combustion is one of the recovery methods in which thermal energy is used to increase oil recovery. The process requires the burning of some reservoir oil in place. The combustion is usually initiated by using an electric burner. Compressed air or oxygen is injected to initiate and maintain controlled combustion. Heat is generated to produce a rise in the reservoir temperature and a reduction in the viscosity of the oil: hence, the oil flows freely towards the production well.

A two-dimensional mathematical model of the in-situ combustion process was developed. The model involves a set of nonlinear partial differential equations, algebraic constraints and phase equilibrium relationships. A computer program was written in FORTRAN 77 to solve the equations involved i.e. to numerically simulate the in-situ combustion process.

The model accounts for the flow of oil, water and gas phases, as well as the formation of a solid coke phase. It has six components and allows four chemical reactions. Heat generation by combustion, heat transfer by conduction, convection within the reservoir and heat loss by conduction to adjacent formations are also included. Additionally, the model includes the injection of heat from a band heater. It allows the model to be used to simulate in-situ combustion tube experiments as well as field projects.

For validation, the results of the present model were compared with results obtained from two other simulators for the base case data in one-dimension.

Furthermore, a two-dimensional run was performed to show the robustness and efficiency of the simulator. It also shows the combustion front movement and the direction it takes.

In order to investigate the grid size sensitivity, several runs were made for one and two-dimensional reservoirs.

The model has the capability to set its own timestep automatically, based on changes of primary variables during the previous timestep, limited by the target change allowed for these variables.

A numerical scheme was developed to solve the system of equations constituting the model. The conservation equations are discretized in finite-difference form. The discretization in time is fully-implicit and central-difference methods combined with one-point upstream weighting are employed for space discretization. The discretization yields seven coupled nonlinear algebraic equations per grid-block. These seven nonlinear algebraic equations can be reduced to six by eliminating the coke concentration equation, because the coke equation contains no interblock flow terms.

The resulting set of nonlinear algebraic equations are solved for each timestep by use of a Newton-Raphson procedure. Each Newton iteration produces an equation of the form

$$A\mathbf{x} = \mathbf{b}, \tag{1}$$

where $\mathbf{x}$ is the Newton update, $\mathbf{b}$ is the current residual of the nonlinear equations and A is the Jacobian matrix. A is large and has a nonsymmetric, sparse structure.

Direct methods (like Gaussian Elimination and LU decomposition) may be inefficient for solving (1) because of the large amount of work and storage involved. Iterative methods can be used instead to obtain an approximate solution.

In this current work we wish to compare the performance of LU decomposition, ORTHOMIN(m) and more recent iterative methods, GMRES(m) and BI-CGSTAB on the model of the in-situ combustion problem.

To increase the convergence rate for the iterative methods a preconditioning and a scaling technique are used.

Acknowledgements

First and foremost I would like to express my sincere gratitude to Mr.F.Brian Ellerby for his constant supervision, interest, encouragement and support during the course of my research.

My gratitude should also be expressed to Dr.Malcolm Greaves from the School of Chemical Engineering for many useful comments.

I would like to acknowledge the School of Mathematical Science of the University of Bath for their excellent computing facilities. I also wish to thank to all the staff of the School of Mathematical Science, for warm friendship and lectures, especially Prof. Alastair Spence and Dr. Ivan Graham.

I am grateful to the Turkish Government, the Higher Education Council of Turkey and the İnönü University for their financial support and scholarship they provided.

I would like to thank to Prof. Sadık Keleş and my colleagues in Turkey for their help in various ways.

I am deeply indebted to my wife, Nisa and my daughter, İrem for being a constant source of love and strength and having the comfort of a real home.

Finally and perhaps most importantly I am extremely grateful to my parents for their continual support and encouragement throughout my education.

Ahmet Refik Bahadır

Bath Oct '94

Contents

Chapter 1

Introduction

1.1 Background of Oil Recovery Operations

Crude oil accumulates over geologic time in porous underground rock formations called reservoirs, where it has been trapped by overlying and adjacent impermeable rock. During the first half of the twentieth century, when new discoveries of oil were made repeatedly, there was little incentive to produce petroleum efficiently. Much was wasted when it was left behind in the reservoir. As petroleum has grown more precious, the petroleum industry has effected curbs to stop such waste by initiating measures to ensure more exacting, more complete recovery.

There are three groups of techniques for the recovery of oil [17];

a) Primary recovery,

b) Secondary recovery,

c) Tertiary recovery.

a) Primary recovery, as the term suggests, is the first method of producing oil from a well. This method makes use of the natural energy stored within the reservoir.

When discovered, a crude oil reservoir contains a mixture of water, oil and gas in the small pore spaces (holes) in the reservoir rock. Initially, the oil is under

considerable pressure, caused by the hydrostatic pressure of the ground water and expansion of volatile components. This pressure can be used to force oil up to the surface.

Eventually, the rate of production from a flowing well tends to decline as the natural drive energy is expended. When this occurs, energy must be added to the reservoir to produce more oil. Hence the secondary phase of oil production begins.

b) Secondary recovery techniques involve the introduction of energy into the reservoir by injecting gas or water under pressure. Separate wells are used for injection and production. The injected fluids maintain reservoir pressure, and displace a portion of the remaining crude oil to the production wells. The usual secondary technique is the injection of water, known as waterflooding.

Unfortunately, not all the oil is produced by these processes. The combined total oil production by conventional primary and secondary recovery processes is generally less than 40 percent of original oil in place. Significant quantities of oil therefore remain behind. In fact, approximately 2,000 billion barrels of the world's oil cannot be recovered by conventional methods [44], [47].

For the last three decades, scientists have been searching for new techniques to recover more oil from reservoirs. These techniques are referred to as tertiary recovery methods.

c) The tertiary recovery is the last period in the history of the reservoir and commences with the introduction of chemical and thermal energy to enhance the production of oil so it has been called as enhanced oil recovery (EOR).

Since the early 1950's a significant amount of laboratory research and field testing has been devoted to developing EOR methods.

There are three main EOR methods [38];

i) Chemical flooding,

ii) Miscible flooding,

iii) Thermal recovery.

i) Chemical Flooding

This method is designed to remove the oil from the pores by injecting chemicals which have a greater attraction for the rock than the oil does.

Chemical methods include polymer flooding, surfactant flooding and alkaline flooding processes.

ii) Miscible Flooding

In this method carbon dioxide, nitrogen or hydrocarbon gases are injected, these dissolve in the oil, reducing its viscosity and causing it to flow easily.

iii) Thermal Recovery

Thermal recovery constitutes an important sector of enhanced oil recovery techniques for many oil fields, particularly where low gravity oils are found or where the oil viscosity is unfavourable for conventional methods. Thermal recovery is currently regarded as the best technique for ultimate oil recovery. Approximately 80 percent of tertiary oil that is produced worldwide is by thermal methods.

The basic principle is to raise the internal energy of the reservoir, reducing the viscosity of the oil, which increases its ability to flow towards producing wells.

The three basic methods known to date are;

1) Hot water injection,

2) Steam injection,

3) In-situ combustion.

1.2 In-Situ Combustion Process

In situ is Latin for "in place". Thus, in-situ combustion is simply the burning of fuel where it exists in a reservoir.

In-situ combustion is another thermal recovery process that is used for recovering more oil from reservoirs already depleted by both primary and secondary recovery operations. It is also called fireflooding.

In this process, oxygen or oxygen-containing gas (usually air) is injected into the reservoir, the crude oil in the reservoir is ignited, and part of that crude is burned in the formation to provide the necessary heat.

When the air flow is large enough, spontaneous combustion may take place or otherwise a downhole heater can be used in the injection well to initiate the combustion. The combustion front travels towards the production well at a rate governed principally by the type and amount of fuel burned, the air injection rate, and the oxygen content of the injection air.

The history of in-situ combustion can be traced back to 1923 when the first serious proposals were put forward, and the method was attempted in the Soviet Union in the 1930's. But it was not until the 1950's that extensive laboratory and field work began [2].

Comprehensive reviews of in-situ combustion field projects have periodically appeared, one of the latest being the article by Chu [11], which assesses the state of the art of in-situ combustion. The World's largest in-situ combustion process is in Romania. This project was started in 1964 and is currently producing approximately 10,500 barrels of oil per day with 600 production wells [8].

Several different operating strategies have been used into initiate and maintain combustion in a reservoir. These are classified into three main types;

a) Dry forward combustion,

b) Wet forward combustion,

c) Reverse combustion.

a) Dry Forward Combustion

This process involves injecting air or oxygen enriched air into the injection well such that a combustion zone will be propagated within the reservoir rock towards the production wells.

b) Wet Forward Combustion

The wet combustion process was developed to improve the efficiency of the in-situ combustion processes.

Water is injected simultaneously or alternatively with air to scavenge the heat from the burned sand, because addition of water enhances the production of steam. Steam moves heat forward much more effectively than combustion gases alone. Parrish and Craig [40] have named this alternate air and water injection method the COFCAW process, the Combination of Forward Combustion and Waterflood. The amount of water injected will determine whether the process will be normal wet, incomplete wet or superwet as classified by Burger and Sahuquet [7].

Field pilot tests and laboratory experiments show that a reservoir subjected to wet forward in-situ combustion can be divided into several distinct zones as illustrated in Figure 1-1 [38].

c) Reverse Combustion

In this process, air is initially injected into a well, which will later be a producing well, to start combustion. After ignition has occurred, the combustion front moves from the producing well towards the injection well and further air injection takes place through a different well.

This process is termed "dry" if no water is injected with the gas, or "wet" if small amounts of water are injected.

But forward combustion is most commonly used and has been operated successfully in several locations.

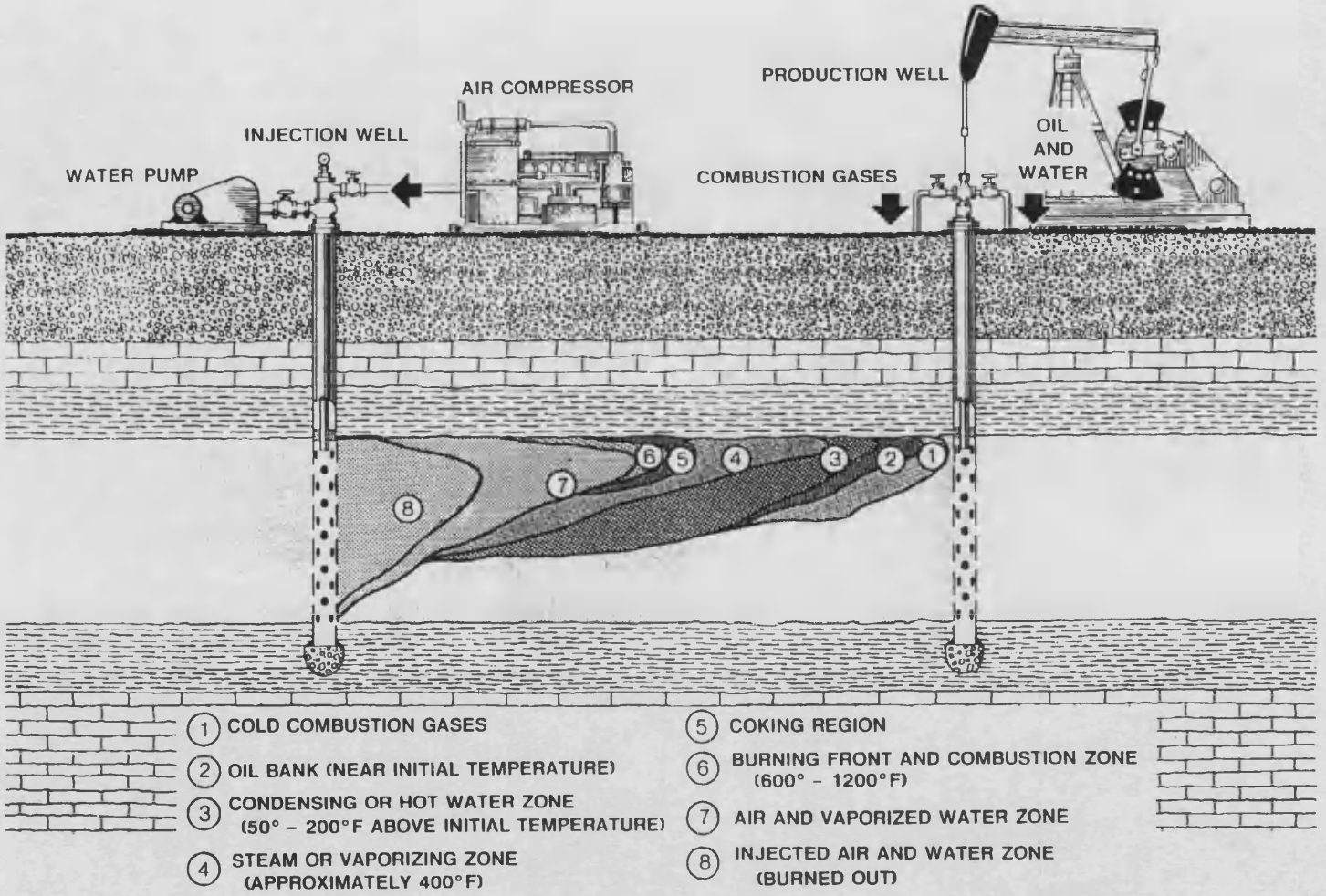


Figure 1-1: Schematic diagram of wet in-situ combustion. (Courtesy of Mr.J.Lindley, U.S.Department of Energy, Bartlesville, Oklahoma.)

1.3 The Simulation of In-Situ Combustion Process

The in-situ combustion recovery process, is extremely complex in nature. It, in fact, includes some aspects of nearly every oil recovery method. The process involves the multiphase flow of gas, oil and water, transfer of heat by conduction and convection and the chemical reactions [1].

Numerical simulation is a powerful tool for studying such a complex reservoir situation. Simulation of petroleum reservoir performance includes the construction and operation of a model whose conduct is similar to the performance of an actual reservoir.

The numerical simulation of in-situ combustion process is intended to predict the effect of different injection rates and well spacings as well as the extent of gravity override [42], heat loss [60], [62], ignition time [6].

Large investments have been based on the results obtained from simulations. A good field project depends on a numerical simulation study as well as laboratory experiment.

The construction of a robust model requires a considerable degree of sophistication and the knowledge of several different areas of specialisation. These are reservoir engineering principles, partial differential equation theory and computer programming. A simulator is an implementation of these disciplines in a computer model.

Figure 1-2 summarizes some major components of an in-situ combustion simulator.

The basic model consists of the nonlinear partial differential continuity equations which govern the multiphase flow of gas, oil and water and the energy equation in the reservoir medium. The simulator is then a collection of computer programs which implement the approximate solution of the differential equations,

which comprise the mathematical model, on a digital machine.

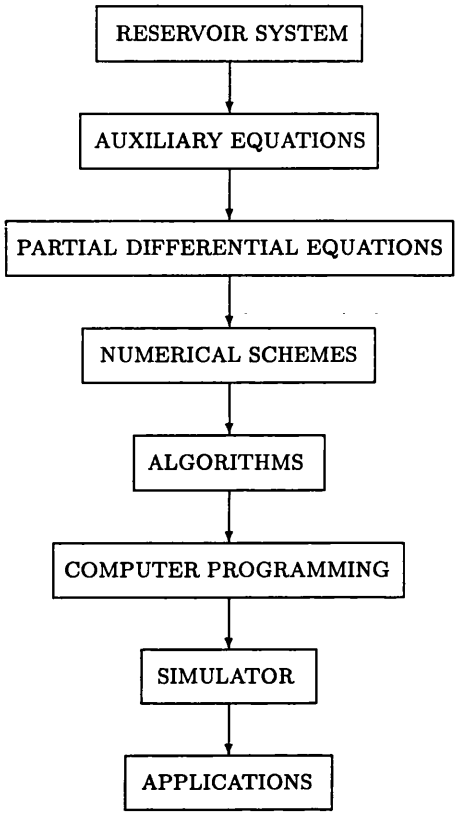


Figure 1-2: What is an in-situ combustion simulator ?

The steps taken in formulating a simulation model consist of developing the following stages [1];

a) Physical model,

b) Mathematical model,

c) Numerical model,

d) Computer model.

a) Physical Model

The physical model is a small scale reproduction of the original problem. This is represented by the process description and possibly the creation of an experimental tube. The experimental tube is a linear elemental model. It represents a small full-scale section of the reservoir and is designed to be a near-adiabatic device.

A series of laboratory experiments was carried out within the School of Chemical Engineering of the University of Bath under Dr.M. Greaves's supervision [32], [54].

b) Mathematical Model

Physical laws of mass and energy conservations will be combined in a complex manner to describe this complicated process. The formulation of the mathematical model is a set of nonlinear partial differential equations and some auxiliary equations. These equations are determined by the number of space dimensions, the number of phases, representation of the chemical reactions and the properties of the reservoir rock and fluids.

c) Numerical Model

The heart of simulation is the solution of a set of partial differential equations. The equations constituting a mathematical model of the reservoir are highly nonlinear and they cannot be solved by analytical methods. Therefore, a numerical technique is necessary to obtain an approximate solution to these equations.

The technique consists in replacing the derivatives by finite-difference approximations. The spatial domain is replaced by a network of discrete points within the domain, and the time domain is divided also into (not necessarily equal) increments. The approximated partial differential equations are then written for each of these discretized points or nodes and the system of algebraic equations (linear or nonlinear) thus developed is solved by a suitable technique, providing an approximate solution to the dependent variables at each of the nodes and at

discrete points in time [41], [48].

Another method for the numerical solution of nonlinear partial differantial equations is the finite-element method. The method basically consists of the representation of spatial domain by an assemblage of subdivisions called finite-elements. A variational principle, such as the principle of minimum potential energy is used to obtain a set of equilibrium equations for each element and the displacements are approximated at each nodal point by employing suitable functions, usually polynomials [33].

Finite-differences are most commonly used at present in this type of numerical model.

d) Computer Model

The computer model is a computer program or a set of programs written to be solve the equations of the numerical model. It constitutes a computer model of the reservoir.

1.4 Literature Review

Many investigators have reported various types of models for in-situ combustion simulation.

Earlier researchers Bailey and Larkin [2], Ramey [43], Baker [3] and Thomas [53] developed mathematical models which considered only certain aspects of the in-situ combustion process, such as, heat transfer with phase change, heat transfer with chemical reaction or three-phase flow. All these models utilised an analytical solution approach.

Numerical models requiring computer solution are more comprehensive than analytic models. To varying degrees, the numerical models include the effects of both fluid and energy flow.

Chu [10] developed the first numerical model to study the propagation of combustion in a radial direction. His model considers the energy effects of vapourization and condensation on the temperature distribution, but neglects the accompanying phase changes by assuming constant fluid saturations.

Even though these models helped workers to understand the different mechanisms involved in the process, there was a need to include all these phenomena in a single mathematical model.

Gottfried [27] presented a comprehensive model which combined heat and mass flow calculations. His model also considered the multiphase flow of gas, oil and water.

The limitations of Gottfried's approach were that gravity and capillary effects were not accounted for, oil was considered to be the only fuel source and no allowance was made for coke formation and oxidation.

The model was solved numerically and the solution procedure relied on an iterative scheme. Since the rate of convergence was very slow, considerable computer time was required to obtain numerical solution.

Smith and Farouq Ali [49] reported the first single phase, two-dimensional in-situ combustion model. Their numerical model accounts for heat generation by a combustion zone, heat transfer by conduction and convection in the reservoir, heat losses by conduction to adjacent formations. Later, Eggenschwiler and Farouq Ali [18] presented an improved version of this model.

El-Khatib [21] developed a one-dimensional, six-component simulator of in-situ combustion. His model accounts for most phenomena associated with the combustion process, but the reaction kinetics are simplified to include only the oxidation of the coke-like fuel. The effects of capillary pressure and gravity are not included.

In 1977, Farouq Ali [23] developed a two-dimensional, four-components, and multiphase simulator. He included the effect of capillary pressure. His model

was more complete than previous models. The solution scheme employed in this model is called IMPES. i.e. iterative solution of the time change in pressure, and explicit solution for time changes in other variables.

Crookston *et al.* [15] described a three-phase, two-dimensional simulator that modeled the most essential features of in-situ combustion. The model equations were solved using IMPES method.

At this time it was becoming clear that the most economical way of modelling a thermally enhanced oil recovery was by using a fully-implicit formulation.

In 1979, A fully-implicit combustion and steamflood simulator, ISCOM, was developed by the Computer Modelling Group of Calgary, Canada [28]. This was by far the most comprehensive model of the time and was able to simulate in-situ combustion and steam injection processes.

Their model accounts four phases, a variable number of oil components, a variable number of chemical reactions, and gravity and capillary pressure terms.

Later, in 1980, several authors, Coats [12], Youngren [65], and Thiez and Lemonnier [52], described finite-difference, multidimensional simulation models for the in-situ combustion process.

Recently, new simulators were produced by Davies [16] and Oklany [39].

Chapter 2

A Two Dimensional In-situ Combustion Simulator

2.1 Description of the Model

A two-dimensional model is developed to simulate the in-situ combustion oil recovery process. It accounts for the flow of gas, oil and water as well as the formation of a solid coke. It has six components: water, heavy oil, light oil, oxygen, inert gas and coke.

Four chemical reactions are accounted for: formation of coke from the heavy hydrocarbon component and the oxidation of coke and both heavy and light hydrocarbon components.

It is designed to handle heat transfer by convection and conduction within the reservoir, conductive heat loss to adjacent strata, and gravity and capillary pressure effects.

The model is additionally capable of simulating the injection of heat from a band heater.

The set of the simulator equations is composed of conservation equations for each component and energy, algebraic constraints, phase equilibrium relationships

and reaction rate equations.

The solution technique employed by the simulator is based on Newton's method.

2.1.1 Mass Conservation Equations

Let us consider a volume element as shown in Figure 2-1.

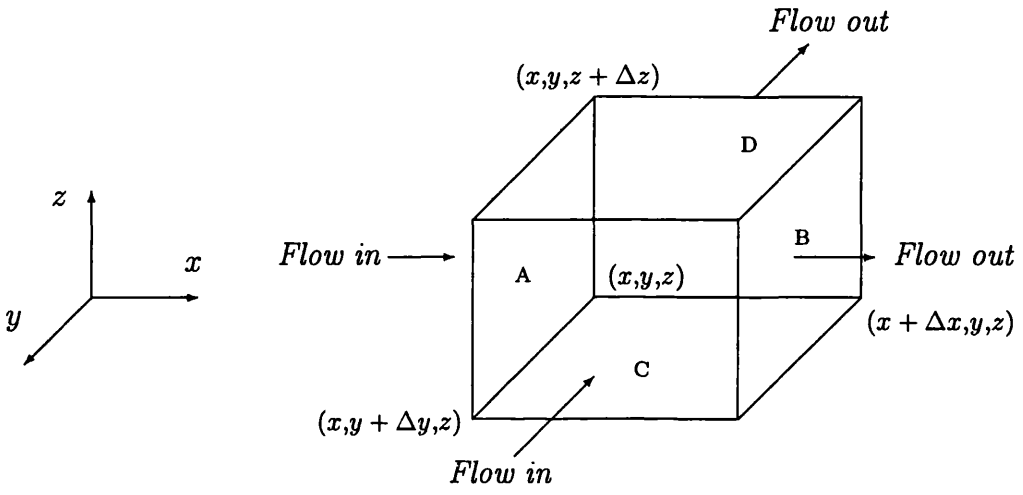


Figure 2-1: Element of volume in two-dimensional flow.

If we consider the single-phase (say, gas phase) flow across the block faces of an element having a volume Δx, Δy and Δz then a mass conservation over the time step Δt for the molar concentration of any component (say, oxygen component) yields:

$$\begin{pmatrix} \textit{Net mass flowing} \\ \textit{in } x-\textit{direction} \end{pmatrix} + \begin{pmatrix} \textit{Net mass flowing} \\ \textit{in } y-\textit{direction} \end{pmatrix} + \begin{pmatrix} \textit{Mass injection or} \\ \textit{production rate} \end{pmatrix}$$

$$= \begin{pmatrix} \textit{Rate of mass} \\ \textit{accumulation} \end{pmatrix}. \tag{2.1}$$

From Figure 2-1,

$$\begin{pmatrix} \textit{Mass inflow across} \\ \textit{the surface } A \end{pmatrix} = \Delta y \Delta z \left(\rho_g V_{g_x}\right)_x$$

and

$$\begin{pmatrix} \textit{Mass outflow across} \\ \textit{the surface } B \end{pmatrix} = \Delta y \Delta z \left(\rho_g V_{g_x}\right)_{x+\Delta x}.$$

Therefore,

$$\begin{pmatrix} \textit{Net mass flowing} \\ \textit{in } x-\textit{direction} \end{pmatrix} = \begin{pmatrix} \textit{Mass inflow across} \\ \textit{the surface } A \end{pmatrix} - \begin{pmatrix} \textit{Mass outflow across} \\ \textit{the surface } B \end{pmatrix}.$$

Hence,

$$\begin{pmatrix} \textit{Net mass flowing} \\ \textit{in } x-\textit{direction} \end{pmatrix} = \Delta y \Delta z \left[\left(\rho_g V_{g_x}\right)_x - \left(\rho_g V_{g_x}\right)_{x+\Delta x}\right] \tag{2.2}$$

and similarly for the y-direction,

$$\begin{pmatrix} \textit{Net mass flowing} \\ \textit{in } y-\textit{direction} \end{pmatrix} = \Delta x \Delta z \left[\left(\rho_g V_{g_y}\right)_y - \left(\rho_g V_{g_y}\right)_{y+\Delta y}\right]. \tag{2.3}$$

Other components of the mass conservation equation are:

$$\begin{pmatrix} \textit{Mass injection or} \\ \textit{production rate} \end{pmatrix} = q\Delta x\Delta y\Delta z, \tag{2.4}$$

where q is the mass rate of injection per unit volume of reservoir (a negative q implies production).

If we consider that the porosity of the reservoir medium is ϕ, the mass contained in the box is $\phi\Delta x\Delta y\Delta z\rho_g S_g$. Then,

$$\begin{pmatrix} \textit{Rate of mass} \\ \textit{accumulation} \end{pmatrix} = \phi\Delta x\Delta y\Delta z\left[\frac{(\rho_g S_g)_{t+\Delta t}-(\rho_g S_g)_t}{\Delta t}\right]. \tag{2.5}$$

Substituting the above terms (2.2)-(2.5) into the total mass conservation equation (2.1) yields:

$$\Delta y\Delta z\left[(\rho_g V_{g_x})_x - (\rho_g V_{g_x})_{x+\Delta x}\right] + \Delta x\Delta z\left[\left(\rho_g V_{g_y}\right)_y - \left(\rho_g V_{g_y}\right)_{y+\Delta y}\right] + q\Delta x\Delta y\Delta z$$

$$= \phi\Delta x\Delta y\Delta z\left[\frac{(\rho_g S_g)_{t+\Delta t} - (\rho_g S_g)_t}{\Delta t}\right].$$

Rearranging and dividing by $(\Delta x\Delta y\Delta z)$ gives,

$$-\frac{1}{\Delta x}\left[(\rho_g V_{g_x})_{x+\Delta x} - (\rho_g V_{g_x})_x\right] - \frac{1}{\Delta y}\left[\left(\rho_g V_{g_y}\right)_{y+\Delta y} - \left(\rho_g V_{g_y}\right)_y\right] + q$$

$$= \frac{1}{\Delta t}\phi\left[(\rho_g S_g)_{t+\Delta t} - (\rho_g S_g)_t\right].$$

Taking the limit as Δx,Δy and Δt go to zero simultaneously:

$$-\frac{\partial}{\partial x}\left(\rho_g V_{g_x}\right)-\frac{\partial}{\partial y}\left(\rho_g V_{g_y}\right)+q=\phi\frac{\partial}{\partial t}\left(\rho_g S_g\right)$$

Where the gas velocity (in x-direction, V_{g_x} and in y-direction, V_{g_y}) is defined by the following form of Darcy's law [13].

$$V_{g_x}=-6.328\frac{K}{\mu_g}\left(\frac{\partial P_g}{\partial x}-\frac{\rho_g M_g}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}\right)$$

$$V_{g_y}=-6.328\frac{K}{\mu_g}\left(\frac{\partial P_g}{\partial y}-\frac{\rho_g M_g}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}\right).$$

The mass conservation equations used in this simulator are for a system of six components. The components and their distribution through the phases are presented in Table 2-1.

Phase	**Component**	**Mole Fraction**
Gas	Water	Y_1
	Heavy oil	Y_2
	Light oil	Y_3
	Inert gas	Y_4
	Oxygen	Y_5
Oil	Heavy oil	X_2
	Light oil	X_3
Water	Water	1
Solid	Coke	1

Table 2-1: Distribution of components.

a) Water Mass Conservation Equation

$$-\frac{\partial}{\partial x}\left(\rho_g V_{g_x} Y_1 + \rho_w V_{w_x}\right) - \frac{\partial}{\partial y}\left(\rho_g V_{g_y} Y_1 + \rho_w V_{w_y}\right) + q_1 + \left(s_3 r_A + s_6 r_B + s_{12} r_D\right)$$

$$= \frac{\partial}{\partial t}\left[\phi\left(\rho_g S_g Y_1 + \rho_w S_w\right)\right]. \tag{2.6}$$

b) Heavy Oil Mass Conservation Equation

$$-\frac{\partial}{\partial x}\left(\rho_g V_{g_x} Y_2 + \rho_o V_{o_x} X_2\right) - \frac{\partial}{\partial y}\left(\rho_g V_{g_y} Y_2 + \rho_o V_{o_y} X_2\right) + q_2 - \left(r_B + r_C\right)$$

$$= \frac{\partial}{\partial t}\left[\phi\left(\rho_g S_g Y_2 + \rho_o S_o X_2\right)\right]. \tag{2.7}$$

c) Light Oil Mass Conservation Equation

$$-\frac{\partial}{\partial y}\left(\rho_g V_{g_x} Y_3 + \rho_o V_{o_x} X_3\right) - \frac{\partial}{\partial y}\left(\rho_g V_{g_y} Y_3 + \rho_o V_{o_y} X_3\right) + q_3 - \left(r_A - s_7 r_C\right)$$

$$= \frac{\partial}{\partial t}\left[\phi\left(\rho_g S_g Y_3 + \rho_o S_o X_3\right)\right]. \tag{2.8}$$

d) Inert Gas Mass Conservation Equation

$$-\frac{\partial}{\partial x}\left(\rho_g V_{g_x} Y_4\right) - \frac{\partial}{\partial y}\left(\rho_g V_{g_y} Y_4\right) + q_4 + \left(s_2 r_A + s_5 r_B + s_9 r_C + s_{11} r_D\right)$$

$$= \frac{\partial}{\partial t}\left[\phi\left(\rho_g S_g Y_4\right)\right]. \tag{2.9}$$

e) Oxygen Mass Conservation Equation

$$-\frac{\partial}{\partial x}\left(\rho_g V_{g_x} Y_5\right) - \frac{\partial}{\partial y}\left(\rho_g V_{g_y} Y_5\right) + q_5 - \left(s_1 r_A + s_4 r_B + s_{10} r_D\right)$$

$$= \frac{\partial}{\partial t}\left[\phi\left(\rho_g S_g Y_5\right)\right]. \tag{2.10}$$

f) Coke Mass Conservation Equation

$$s_8 r_C - r_D = \frac{\partial C_c}{\partial t}. \tag{2.11}$$

2.1.2 Energy Conservation Equation

The energy conservation equation contains conduction, convection, reaction, injection/production, band heater and heat loss terms.

If we consider again the stationary volume element in Figure 2-1, a heat balance over this element gives,

$$\begin{pmatrix} \textit{Net rate of heat} \\ \textit{transfer by conduction} \\ \textit{in } x-\textit{direction} \end{pmatrix} + \begin{pmatrix} \textit{Net rate of heat} \\ \textit{transfer by convection} \\ \textit{in } x-\textit{direction} \end{pmatrix}$$

$$+ \begin{pmatrix} \textit{Net rate of heat} \\ \textit{transfer by conduction} \\ \textit{in } y-\textit{direction} \end{pmatrix} + \begin{pmatrix} \textit{Net rate of heat} \\ \textit{transfer by convection} \\ \textit{in } y-\textit{direction} \end{pmatrix}$$

$$+\begin{pmatrix} \textit{Heat injection or} \\ \textit{production rate} \end{pmatrix}+\begin{pmatrix} \textit{Heat generation} \\ \textit{by reaction} \end{pmatrix}+\begin{pmatrix} \textit{Heat injection} \\ \textit{by bandheater} \end{pmatrix}$$

$$-\begin{pmatrix} \textit{Heat loss} \\ \textit{to formation} \end{pmatrix}=\begin{pmatrix} \textit{Rate of heat} \\ \textit{accumulation} \end{pmatrix}.$$

The analysis of the energy conservation equation is similar to the analysis of mass conservation equation which was given in Section 2.1.1.

The energy conservation equation is,

$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial T}{\partial x}\right)-\frac{\partial}{\partial x}\left(\rho_g h_g V_{g_x}+\rho_o h_o V_{o_x}+\rho_w h_w V_{w_x}\right)$$

$$+\frac{\partial}{\partial y}\left(\lambda\frac{\partial T}{\partial y}\right)-\frac{\partial}{\partial y}\left(\rho_g h_g V_{g_y}+\rho_o h_o V_{o_y}+\rho_w h_w V_{w_y}\right)$$

$$+\left(q_g h^*{}_g+q_o h^*{}_o+q_w h^*{}_w\right)+\left(H_A r_A+H_B r_B+H_C r_C+H_D r_D\right)+H_{inj}-H_{loss}$$

$$=\frac{\partial}{\partial t}\left[C_c U_c+(1-\phi)\,\rho_r U_r+\phi\left(S_g\rho_g U_g+S_o\rho_o U_o+S_w\rho_w U_w\right)\right]. \qquad (2.12)$$

Heat injected by the band heater is obtained by

$$H_{inj}=\eta(T_B-T),$$

where η is band heater constant.

The heat loss rate is calculated using the same method as in [60]. This is a semi-analytical method and uses a fitting function for temperature profile into the cap rock or base rock:

$$T_{cap-rock}(t,z) = (T_{cell} + \alpha z + \beta z^2)e^{-z/l}.$$

The equation of heat flow, say, into the cap rock is:

$$\frac{\partial T_{cap-rock}}{\partial t} = \kappa \frac{\partial^2 T_{cap-rock}}{\partial z^2},$$

where $T_{cap-rock}$ is the cap rock temperature, t is time, z is distance into the cap rock, T_{cell} is the grid-block temperature, α and β are parameters determined by the analysis, κ is thermal diffusivity, and l is a diffusion length where $l = \frac{\sqrt{\kappa t}}{2}$.

2.1.3 Chemical Reaction Rate Equations

Reaction Equations:

During in-situ combustion a number of chemical reactions take place. Four chemical reactions are incorporated into the model, and they are the same reactions which Crookston *et al.* [15] include in their model.

1. Oxidation of light oil component:

$$(Light\ oil) + s_1(O_2) \xrightarrow{r_A} s_2(CO_x) + s_3(H_2O).$$

2. Oxidation of heavy oil component:

$$(Heavy\ oil) + s_4(O_2) \xrightarrow{r_B} s_5(CO_x) + s_6(H_2O).$$

3. Thermal cracking of heavy oil component to form light oil component, coke and inert gas:

$$(Heavy\ oil) \xrightarrow{r_C} s_7(Light\ oil) + s_8(Coke) + s_9(Inert\ gas).$$

4. Oxidation of coke formed by cracking:

$$(Coke) + s_{10}(O_2) \xrightarrow{r_D} s_{11}(CO_x) + s_{12}(H_2O).$$

Reaction Rates:

The kinetic reaction rate expressions are used for the above reactions are;

$$r_A = A_A e^{(-E_A/RT)} Y_5 P_g (\phi \rho_o S_o X_3),$$

$$r_B = A_B e^{(-E_B/RT)} Y_5 P_g (\phi \rho_o S_o X_2),$$

$$r_C = A_C e^{(-E_C/RT)} (\phi \rho_o S_o X_2) \left[1 - \left(\frac{C_c}{C_{c_{max}}}\right)^5\right], \tag{2.13}$$

$$r_D = A_D e^{(-E_D/RT)} Y_5 P_g C_c. \tag{2.14}$$

2.1.4 Algebraic Constraints and Phase Equilibrium Ratios

In addition to the differential equations, certain auxiliary relations must be provided to complete the description of the mathematical model.

Saturation Constraint :

The sum of the volumes of the three phases must always equal to the pore volume at any point in the system. Hence,

$$S_g + S_o + S_w = 1. \tag{2.15}$$

Obviously, any one saturation can always be expressed in terms of the other two.

Oil Mole Fraction Constraint :

In each phase, the mass fractions must add up to 1.

$$X_2 + X_3 = 1. \tag{2.16}$$

Gas Mole Fraction Constraint :

$$Y_1 + Y_2 + Y_3 + Y_4 + Y_5 = 1. \tag{2.17}$$

Phase Equilibrium Ratios :

$$Y_1 = K_1, \tag{2.18}$$

$$Y_2 = K_2 X_2, \tag{2.19}$$

$$Y_3 = K_3 X_3. \tag{2.20}$$

Capillary Pressure Relations :

$$P_{cgo} = P_g - P_o, \tag{2.21}$$

$$P_{cow} = P_o - P_w. \tag{2.22}$$

Production/Injection Terms :

The production rates for the three phases are expressed as

$$q_g = \frac{\gamma K \Delta z}{\Delta x \Delta y \Delta z} \frac{\rho_g K_{rg}}{\mu_g} (P_g - P_{prod}),$$

$$q_o = \frac{\gamma K \Delta z}{\Delta x \Delta y \Delta z} \frac{\rho_o K_{ro}}{\mu_o} (P_o - P_{prod}),$$

$$q_w = \frac{\gamma K \Delta z}{\Delta x \Delta y \Delta z} \frac{\rho_w K_{rw}}{\mu_w} (P_w - P_{prod}).$$

The Phase Velocity Equations :

The phase velocities are calculated using Darcy's Law as follows [13].

$$V_{g_x} = -6.328 \frac{K K_{rg}}{\mu_g} \left(\frac{\partial P_g}{\partial x} - \frac{\rho_g M_g}{144} \frac{\mathbf{g}}{g_c} \frac{\partial D}{\partial x} \right), \tag{2.23}$$

$$V_{g_y} = -6.328 \frac{K K_{rg}}{\mu_g} \left(\frac{\partial P_g}{\partial y} - \frac{\rho_g M_g}{144} \frac{\mathbf{g}}{g_c} \frac{\partial D}{\partial y} \right), \tag{2.24}$$

$$V_{o_x} = -6.328 \frac{K K_{ro}}{\mu_o} \left(\frac{\partial P_o}{\partial x} - \frac{\rho_o M_o}{144} \frac{\mathbf{g}}{g_c} \frac{\partial D}{\partial x} \right),$$

$$V_{o_y} = -6.328 \frac{K K_{ro}}{\mu_o} \left(\frac{\partial P_o}{\partial y} - \frac{\rho_o M_o}{144} \frac{\mathbf{g}}{g_c} \frac{\partial D}{\partial y} \right),$$

$$V_{w_x} = -6.328 \frac{KK_{rw}}{\mu_w} \left(\frac{\partial P_w}{\partial x} - \frac{\rho_w M_w}{144} \frac{\mathbf{g}}{g_c} \frac{\partial D}{\partial x} \right),$$

$$V_{w_y} = -6.328 \frac{KK_{rw}}{\mu_w} \left(\frac{\partial P_w}{\partial y} - \frac{\rho_w M_w}{144} \frac{\mathbf{g}}{g_c} \frac{\partial D}{\partial y} \right).$$

2.1.5 Physical Properties

The numerical values for the constants in the following formulae are specified in the data file given as Appendix D. They are taken from [12], [15] and [28].

Viscosities :

Gas Viscosity

$$\mu_g = Y_1 * \mu_1 + Y_2 * \mu_2 + Y_3 * \mu_3 + Y_4 * \mu_4 + Y_5 * \mu_5,$$

where

$$\mu_1 = AGV1 * T^{BGV1},$$

$$\mu_2 = AGV2 * T^{BGV2},$$

$$\mu_3 = AGV3 * T^{BGV3},$$

$$\mu_4 = AGV4 * T^{BGV4},$$

$$\mu_5 = AGV5 * T^{BGV5}.$$

Oil Viscosity

$$\mu_o = {\mu_2}^{X_2} * {\mu_3}^{X_3},$$

where

$$\mu_2 = AOV2 * e^{\frac{BOV2}{T}}$$

and

$$\mu_3 = AOV3 * e^{\frac{BOV3}{T}}.$$

Water Viscosity

$$\mu_w = \frac{AWV}{BWV + CWV * T + DWV * T^2}.$$

Densities :

Gas Density

$$\rho_g = \frac{P_g}{RT}.$$

Oil Density

$$\rho_o = \frac{1}{X_2 * \rho_2 + X_3 * \rho_3},$$

where

$$\rho_2 = AOD2 * [1 - BOD2 * (P_o - P_{ref})][1 + COD2 * (T - T_{ref})]$$

and

$$\rho_3 = AOD3 * [1 - BOD3 * (P_o - P_{ref})][1 + COD3 * (T - T_{ref})].$$

Water Density

$$\rho_w = AWD * [1 + BWD * (P_w - P_{ref}) - CWD * (T - T_{ref})].$$

The Equilibrium K-values :

$$K_1 = \frac{1}{P_g}\left(\frac{T - AK1}{BK1}\right)^{CK1} \frac{S_w}{S_w + \varepsilon},$$

$$K_2 = \frac{1}{P_g} e^{\left(\frac{AK2+BK2}{T-CK2}\right)} \frac{S_o}{S_o + \varepsilon},$$

$$K_3 = \frac{1}{P_g} e^{\left(\frac{AK3+BK3}{T-CK3}\right)},$$

where ε is a small number of order 10^{-4}. These expressions are the same as these used by Crookston *et al.* [15].

Relative Permeabilities :

At any given position in the reservoir, relative permeabilities are taken to be function of saturation alone. The three phase relative permeabilities are same as in [15], [39], [58].

$$K_{rg} = \begin{cases} K_{rgro} * \left(\frac{S_g - S_{gc}}{1 - S_{wc} - S_{gc} - S_{org}}\right)^{Z_g}, & \text{if } S_g > S_{gc}, \\ 0, & \text{otherwise,} \end{cases}$$

$$K_{rw} = \begin{cases} K_{rwro} * \left(\frac{S_w - S_{wc}}{1 - S_{wc} - S_{orw}}\right)^{Z_w}, & \text{if } S_w > S_{wc}, \\ 0, & \text{otherwise,} \end{cases}$$

$$K_{ro} = K_{rocw} * \left[\left(\frac{K_{row}}{K_{rocw}} + K_{rw}\right)\left(\frac{K_{rog}}{K_{rocw}} + K_{rg}\right) - K_{rg} - K_{rw}\right],$$

where

$$K_{rog} = K_{rocw} * \left(\frac{1 - S_g - S_{wc} - S_{org}}{1 - S_{gc} - S_{wc} - S_{org}} \right)^{Z_{og}}$$

and

$$K_{row} = K_{rocw} * \left(\frac{1 - S_w - S_{orw}}{1 - S_{wc} - S_{orw}} \right)^{Z_{ow}}.$$

Capillary Pressures :

$$P_{cgo} = CPG1 * S_g + CPG2,$$

$$P_{cow} = CPW1 * S_w + CPW2.$$

Molecular Weights :

$$M_g = M_1 * Y_1 + M_2 * Y_2 + M_3 * Y_3 + M_4 * Y_4 + M_5 * Y_5,$$

$$M_o = M_2 * X_2 + M_3 * X_3,$$

$$M_w = M_1.$$

Enthaplies and Internal Energies :

Gas Enthalpy

$$h_g = \int_{T_{ref}}^{T} C_{p_g} dT,$$

where

$$C_{p_g} = GE1 * Y_1 + GE2 * Y_2 + GE3 * Y_3 + GE4 * Y_4 + GE5 * Y_5.$$

Oil Enthalpy

$$h_o = h_{o_{oil}} - h_{o_{vap}},$$

where

$$h_{o_{oil}} = \int_{T_{ref}}^{T} C_{p_{o_{oil}}} dT,$$

and

$$C_{p_{o_{oil}}} = GE2 * X_2 + GE3 * X_3$$

also

$$h_{o_{vap}} = \begin{cases} OEV1 * (T_{co} - T)^{0.38}, & \text{if } T_{co} > T, \\ 0, & \text{otherwise,} \end{cases}$$

and

$$T_{co} = T_{c2} * X_2 + T_{c3} * X_3.$$

Water Enthalpy

$$h_w = h_{w_{gas}} - h_{w_{vap}},$$

where

$$h_{w_{gas}} = \int_{T_{ref}}^{T} GE1 dT$$

and

$$h_{w_{vap}} = \begin{cases} WEV1 * (T_{cw} - T)^{0.38}, & \text{if } T_{cw} > T, \\ 0, & \text{otherwise.} \end{cases}$$

Gas Internal Energy

$$U_g = h_g - \frac{P_g}{\rho_g}.$$

Oil Internal Energy

$$U_o = h_o - \frac{P_o}{\rho_o}.$$

Water Internal Energy

$$U_w = h_w - \frac{P_w}{\rho_w}.$$

Coke Internal Energy

$$U_c = CIE * (T - T_{ref}).$$

Rock Internal Energy

$$U_r = RIE * (T - T_{ref}).$$

2.1.6 Boundary Condations

Boundary condations are considered to be no flow of mass or energy accross the reservoir boundaries. This is done by setting the depending variables to zero in the boundary grid blocks.

Chapter 3

Numerical Solution of the Conservation Equations

The equations (2.6)-(2.12) are nonlinear partial differential equations. It is impossible to solve these equations analytically. Therefore, the numerical technique of *finite-differences*, will be used.

The reservoir is divided into a number of grid-blocks and implicit finite-difference representations of the six partial differential equations are evaluated in each grid-block. These equations are discretized using central-differences with combined upstream weighting for the spatial variable and backward-differences in time [1].

The resulting system of nonlinear finite-difference equations are solved by a variant of the Newton-Raphson method.

The linear system of equations implicit in the case of Newton's method was solved by using *direct* or *iterative* methods at each step of the Newton's iteration.

3.1 Discretization of the Partial Differential Equations

3.1.1 Spatial Discretization

The numerical solution of partial differential equations by finite-differences involves replacing the partial derivatives by finite-difference quotients. Then, instead of obtaining a continuous solution we obtain an approximate solution at a discrete set of grid-points at distinct times. In this section, we discuss the replacement of the spatial derivatives.

A discrete set of grid-blocks in the xy-plane is obtained by use of a grid system to divide the solution domain rectangle, as in Figure 3-1. There are N_x grid-blocks in the x direction and N_y grid-blocks in the y direction. The subscript i is used to identify columns of grid-blocks and the subscript j to identify rows. Each point (x_i, y_j), also called point (i, j), lies at the centre of a grid-block. Double subscripts were used to indicate positions within the grid. Thus, for example, $U_{i,j}$, refers to the value of U at point (i, j), and more generally to the average value of U within grid-block(i, j).

As seen in Figure 3-2, the locations of the grid-blocks are defined by locations of their boundaries:

$$x_{1/2}, x_{3/2}, \ldots, x_{N_x+1/2}$$

and

$$y_{1/2}, y_{3/2}, \ldots, y_{N_y+1/2}.$$

The centres of the grid-blocks then satisfy

$$x_i = \frac{1}{2}(x_{i-1/2} + x_{i+1/2})$$

and

$$y_i = \frac{1}{2}(y_{i-1/2} + y_{i+1/2}).$$

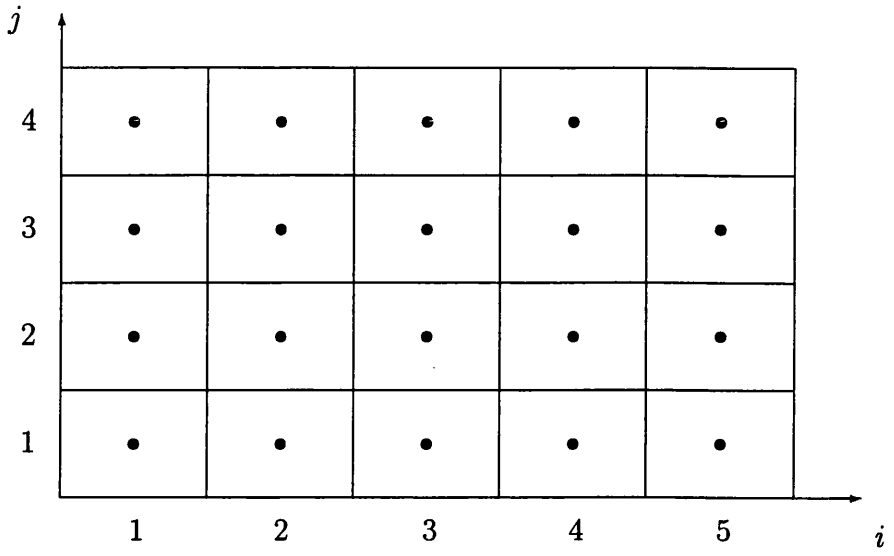


Figure 3-1: Block centred grid system ($N_x = 5, N_y = 4$).

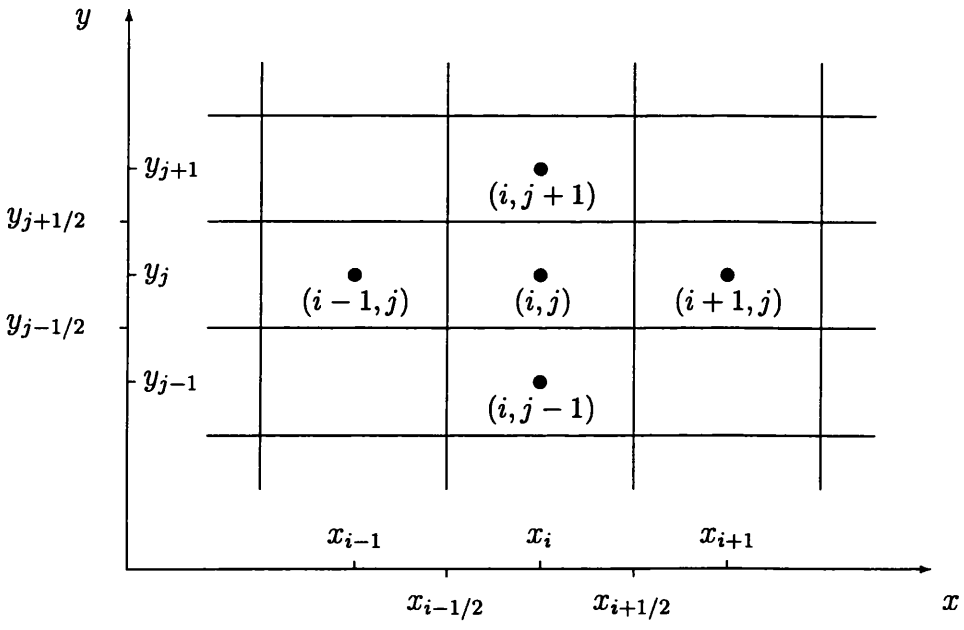


Figure 3-2: Details of block centred grid system.

We can now begin the replacement of the space derivatives. For simplicity, we will consider just one of the seven partial differential equations, namely the oxygen mass conservation equation (2.10).

After substituting the phase velocity equations (2.23) and (2.24) into the oxygen mass conservation equation, equation (2.10) then becomes

$$\frac{\partial}{\partial x}\left(\mathcal{U}\frac{\partial P_g}{\partial x}\right) - \frac{\partial}{\partial x}\mathcal{V}_x + \frac{\partial}{\partial y}\left(\mathcal{U}\frac{\partial P_g}{\partial y}\right) - \frac{\partial}{\partial y}\mathcal{V}_y + \mathcal{R} = \frac{\partial}{\partial t}\mathcal{W}, \tag{3.1}$$

where

$$\mathcal{U} = 6.328\frac{\rho_g K K_{rg} Y_5}{\mu_g},$$

$$\mathcal{V}_x = \frac{6.328}{144}\frac{\rho_g^2 K K_{rg} Y_5 M_g}{\mu_g}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x},$$

$$\mathcal{V}_y = \frac{6.328}{144}\frac{\rho_g^2 K K_{rg} Y_5 M_g}{\mu_g}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y},$$

$$\mathcal{R} = q_5 - (s_1 r_A + s_4 r_B + s_{10} r_D),$$

and

$$\mathcal{W} = \phi\left(\rho_g S_g Y_5\right).$$

Consider the first x derivative in the equation (3.1). Let $\Delta x = x_{i+1} - x_i$. Then we can approximate it at point (x_i, y_j) by using *central-differences* i.e.,

$$\frac{\partial}{\partial x}\left(\mathcal{U}\frac{\partial P_g}{\partial x}\right) \approx \frac{1}{\Delta x}\left[\left(\mathcal{U}\frac{\partial P_g}{\partial x}\right)_{i+1/2,j} - \left(\mathcal{U}\frac{\partial P_g}{\partial x}\right)_{i-1/2,j}\right].$$

In turn, we can make approximations such as

$$\left(\frac{\partial P_g}{\partial x}\right)_{i+1/2,j} \approx \frac{(P_g)_{i+1,j} - (P_g)_{i,j}}{\Delta x}$$

and

$$\left(\frac{\partial P_g}{\partial x}\right)_{i-1/2,j} \approx \frac{(P_g)_{i,j} - (P_g)_{i-1,j}}{\Delta x},$$

where $(P_g)_{i,j}$ is the approximate, or numerical, solution for P_g at point (i,j).

Upstream weighting was used to evaluate $(\mathcal{U})_{i\pm1/2,j}$. That is,

$$(\mathcal{U})_{i+1/2,j} = (\mathcal{U})_{i,j}$$

and

$$(\mathcal{U})_{i-1/2,j} = (\mathcal{U})_{i-1,j}\,.$$

Then

$$\frac{\partial}{\partial x}\left(\mathcal{U}\frac{\partial P_g}{\partial x}\right) \approx \frac{1}{\Delta x}\left\{\mathcal{U}_{i,j}\left[\frac{(P_g)_{i+1,j} - (P_g)_{i,j}}{\Delta x}\right] - \mathcal{U}_{i-1,j}\left[\frac{(P_g)_{i,j} - (P_g)_{i-1,j}}{\Delta x}\right]\right\}.$$

Since the flow is in the direction of increasing x this means that the coefficients are evaluated at known points. Using the upstream weighting for the second x derivative in the equation (3.1), we obtain

$$\frac{\partial}{\partial x}\mathcal{V}_x \approx \frac{(\mathcal{V}_x)_{i,j} - (\mathcal{V}_x)_{i-1,j}}{\Delta x}.$$

Similarly, for the y derivatives,

$$\frac{\partial}{\partial y}\left(\mathcal{U}\frac{\partial P_g}{\partial y}\right) \approx \frac{1}{\Delta y}\left\{\mathcal{U}_{i,j}\left[\frac{(P_g)_{i,j+1} - (P_g)_{i,j}}{\Delta y}\right] - \mathcal{U}_{i,j-1}\left[\frac{(P_g)_{i,j} - (P_g)_{i,j-1}}{\Delta y}\right]\right\}$$

and

$$\frac{\partial}{\partial y}\mathcal{V}_y \approx \frac{(\mathcal{V}_y)_{i,j} - (\mathcal{V}_y)_{i,j-1}}{\Delta y}.$$

3.1.2 Time Discretization

To complete the discretization, we also replace each time derivative by a corresponding difference quotient. Time is divided into discrete points: $t^0, t^1, t^2, \ldots, t^n$, $t^{n+1}, \ldots$. (Note that we used superscripts to indicate the time level and t^0 indicates $Time = 0$.) Assume that we have, at time t^n, a solution for each dependent variable at each point. A numerical procedure consists, then, of a technique for generating a solution for each dependent variable at each point at the next time, t^{n+1}. Clearly, starting with the initial condition at time zero, repeated application of the numerical procedure will generate the solution at each discrete point in space and time, (x_i, y_j, t^n), for the period of interest.

Let $\Delta t = t^{n+1} - t^n$. Then the time derivative on the right hand side of equation (3.1), may be approximated at grid-point (i, j) by

$$\frac{\partial}{\partial t}\mathcal{W} \approx \frac{\mathcal{W}_{i,j}^{n+1} - \mathcal{W}_{i,j}^{n}}{\Delta t}.$$

Substituting the numerical approximations for the spatial and time derivatives into the differential equation (3.1) gives the following difference equation:

$$\frac{1}{\Delta x}\left\{\mathcal{U}_{i,j}^{n+1}\left[\frac{(P_g)_{i+1,j}^{n+1} - (P_g)_{i,j}^{n+1}}{\Delta x}\right] - \mathcal{U}_{i-1,j}^{n+1}\left[\frac{(P_g)_{i,j}^{n+1} - (P_g)_{i-1,j}^{n+1}}{\Delta x}\right]\right\}$$

$$+\frac{1}{\Delta y}\left\{\mathcal{U}_{i,j}^{n+1}\left[\frac{(P_g)_{i,j+1}^{n+1} - (P_g)_{i,j}^{n+1}}{\Delta y}\right] - \mathcal{U}_{i,j-1}^{n+1}\left[\frac{(P_g)_{i,j}^{n+1} - (P_g)_{i,j-1}^{n+1}}{\Delta y}\right]\right\}$$

$$-\frac{(\mathcal{V}_x)_{i,j}^{n+1} - (\mathcal{V}_x)_{i-1,j}^{n+1}}{\Delta x} - \frac{(\mathcal{V}_y)_{i,j}^{n+1} - (\mathcal{V}_y)_{i,j-1}^{n+1}}{\Delta y} + \mathcal{R}_{i,j}^{n+1} = \frac{\mathcal{W}_{i,j}^{n+1} - \mathcal{W}_{i,j}^{n}}{\Delta t}.$$

The superscripts that have been used on the left hand side of this equation

indicate an *implicit finite-difference* formulation. In fact this is the standard Backward Euler method.

Similar expansion and discretizations were used for the other conservation equations. Complete discretization of all conservation equations have been given in Appendix A.

3.2 Embedding of Coke Conservation Equation

Examination of the coke conservation equation indicates that a simplification is possible in the model formulation. Coke concentration, C_c is not a function of variables in adjacent blocks, therefore C_c can be determined without involving other equations.

Let us consider the equation (A.6)

$$(s_8 r_C - r_D)^{n+1} = \frac{1}{\Delta t}\left(C_c^{n+1} - C_c^n\right). \tag{3.2}$$

From the equations (2.13) and (2.14)

$$r_C{}^{n+1} = \left\{A_C e^{(-E_C/RT)}(\phi\rho_o S_o X_2)\left[1 - \left(\frac{C_c}{C_{c_{max}}}\right)^5\right]\right\}^{n+1} \tag{3.3}$$

$$r_D{}^{n+1} = \left[A_D e^{(-E_D/RT)} Y_5 P_g C_c\right]^{n+1}. \tag{3.4}$$

Substitution of (3.3) and (3.4) into (3.2) and collecting terms yields,

$$C_c^{n+1} = C_c^n + \left\{s_8 A_C e^{(-E_C/RT)}(\phi\rho_o S_o X_2)\left[1 - \left(\frac{C_c}{C_{c_{max}}}\right)^5\right]\right\}^{n+1} \Delta t$$

$$- \left[A_D e^{(-E_D/RT)} Y_5 P_g C_c\right]^{n+1} \Delta t.$$

Therefore, we have (on rearrangement),

$$\mathcal{G}(C_c^{n+1}) = 0. \tag{3.5}$$

This is an implicit nonlinear algebraic equation for C_c^{n+1}. The model uses Newton's method, which is detailed in the next section, to solve equation (3.5) for the root C_c^{n+1}.

3.3 Functional Dependency

The functional dependencies of the physical properties allowed in the simulator are summarised in Table 3-1.

Variable	Functional Dependency
μ_g	$T, Y_1, Y_2, Y_3, Y_4, Y_5$
μ_o	T, X_2, X_3
μ_w	T
ρ_g	T, P_g
ρ_o	T, P_o, X_2, X_3
ρ_w	T, P_w
K_1	T, P_g, S_w
K_2	T, P_g, S_o
K_3	T, P_g
K_{rg}	S_g
K_{ro}	S_g, S_w
K_{rw}	S_w
P_{cgo}	S_g
P_{cow}	S_w

(Cont.)

(Cont.)

M_g	Y_1, Y_2, Y_3, Y_4, Y_5
M_o	X_2, X_3
h_g	$T, Y_1, Y_2, Y_3, Y_4, Y_5$
h_o	T, X_2, X_3
h_w	T
U_g	$T, P_g, Y_1, Y_2, Y_3, Y_4, Y_5$
U_o	T, P_o, X_2, X_3
U_w	T, P_w
U_c	T
U_r	T

Table 3-1: Functional dependencies of physical properties.

Equations (2.15)-(2.22) are used to eliminate some of the variables in the following manner (Table 3-2).

Equations Used	**Eliminated Variables**
(2.15)	S_g
(2.16)	X_3
(2.17)	Y_4
(2.18)	Y_1
(2.19)	Y_2
(2.20)	Y_3
(2.21)	P_o
(2.22)	P_w

Table 3-2: Elimination of some variables.

3.4 Automatic Timestep Selection

The size of the timestep has a great effect on the simulation results. Too large a timestep incurs large time truncation errors. This reduces the quality of the estimation of the reservoir equation nonlinearity. Too small a timestep increases computer time and cost.

Consequently, an automatic timestep selection algorithm is a necessary feature for an implicit in-situ combustion simulator.

The model has the capability to set timesteps automatically. In this method the timestep is estimated by comparing the maximum changes of primary variables during the previous timestep with the target change allowed for these variables. This time step selection method has been found very successful in the in-situ combustion simulation ISCOM [28], [45].

The timestep is calculated from the formula:

$$\Delta t^{n+1} = \Delta t^n min_{\psi_i} \left(\frac{2\Delta\psi}{\Delta\psi + \delta\psi_i} \right), \qquad i = 1, 2, \ldots, N_{gb},$$

where ψ stands for each primary variable, $P_g, S_o, S_w, T, X_2, Y_5$, $\Delta\psi$ is the target change allowed in ψ over a timestep and $\delta\psi$ is the maximum change in ψ during the previous timestep. If the new timestep is greater then Δt_{max} then Δt_{max} is taken as a new timestep. $\Delta\psi$ and Δt_{max} are read from the data file.

3.5 Solution of the System of Non-Linear Equations

3.5.1 The Newton-Raphson Method

The Newton-Raphson (or simply Newton's) method is the best known, and possibly the most widely used, technique for finding the roots of nonlinear alge-

braic equations

$$f(x) = 0. \tag{3.6}$$

This method is based on a Taylor series expansion of the linear function $f(x)$ around an initial estimate $x^{(0)}$ of the root:

$$f(x) = f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)}) + \frac{1}{2!}f''(x^{(0)})(x - x^{(0)})^2 + \frac{1}{3!}f'''(x^{(0)})(x - x^{(0)})^3 + \cdots \tag{3.7}$$

Since what is being sought is the value of x which forces the function $f(x)$ to assume a zero value, the left side of equation (3.7) is set to zero and the resulting equation is solved for x. However, the right hand side is an infinite series. Therefore a finite number of terms are retained and the remaining terms are deleted. Retaining only the first two terms on the right hand side of the Taylor series is equivalent to linearising the function $f(x)$. This operation results in

$$x = x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})} \tag{3.8}$$

i.e., the value of x is calculated from $x^{(0)}$ by correcting this initial guess by subtracting $f(x^{(0)})/f'(x^{(0)})$.

Since the Taylor series was truncated, retaining only two terms, the new value x will not satisfy equation (3.6). We will designate this value as $x^{(1)}$ and reapply the Taylor series linearization at $x^{(1)}$ to obtain $x^{(2)}$.

Repetitive application of this step converts the equation (3.8) to the Newton's iterative formula:

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, \qquad k = 0, 1, 2, 3, \ldots.$$

Now let us consider the set of nonlinear algebraic equations:

$$f_1(x_1, x_2, \ldots, x_n) = 0,$$

$$f_2(x_1, x_2, \ldots, x_n) = 0, \tag{3.9}$$
$$\vdots$$
$$f_n(x_1, x_2, \ldots, x_n) = 0,$$

which we can write in vector form as;

$$F(\mathbf{x}) = \mathbf{0}. \tag{3.10}$$

We wish to find the set of x_i satisfying (3.9), we proceed in exactly the same way. We expand (3.9) in a Taylor series about the $\mathbf{x^{(0)}}$.

$$f_1(\mathbf{x}) = f_1(\mathbf{x}^{(0)}) + \frac{\partial f_1(\mathbf{x}^{(0)})}{\partial x_1}(x_1 - x_1^{(0)}) + \frac{\partial f_1(\mathbf{x}^{(0)})}{\partial x_2}(x_2 - x_2^{(0)}) + \cdots + \frac{\partial f_1(\mathbf{x}^{(0)})}{\partial x_n}(x_n - x_n^{(0)}) + \cdots,$$

$$f_2(\mathbf{x}) = f_2(\mathbf{x}^{(0)}) + \frac{\partial f_2(\mathbf{x}^{(0)})}{\partial x_1}(x_1 - x_1^{(0)}) + \frac{\partial f_2(\mathbf{x}^{(0)})}{\partial x_2}(x_2 - x_2^{(0)}) + \cdots + \frac{\partial f_2(\mathbf{x}^{(0)})}{\partial x_n}(x_n - x_n^{(0)}) + \cdots, \tag{3.11}$$

$$\vdots$$

$$f_n(\mathbf{x}) = f_n(\mathbf{x}^{(0)}) + \frac{\partial f_n(\mathbf{x}^{(0)})}{\partial x_1}(x_1 - x_1^{(0)}) + \frac{\partial f_n(\mathbf{x}^{(0)})}{\partial x_2}(x_2 - x_2^{(0)}) + \cdots + \frac{\partial f_n(\mathbf{x}^{(0)})}{\partial x_n}(x_n - x_n^{(0)}) + \cdots.$$

Setting the left sides of (3.11) to zero and drop the second order terms, we obtain in matrix form

$$J(\mathbf{x}^{(0)})(\mathbf{x} - \mathbf{x}^{(0)}) = -F(\mathbf{x}^{(0)}), \tag{3.12}$$

where $J(\mathbf{x})$ is the Jacobian matrix of $F(\mathbf{x})$ as follows

$$J(\mathbf{x}^{(0)}) = \begin{pmatrix} \frac{\partial f_1(\mathbf{x}^{(0)})}{\partial x_1} & \frac{\partial f_1(\mathbf{x}^{(0)})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x}^{(0)})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x}^{(0)})}{\partial x_1} & \frac{\partial f_2(\mathbf{x}^{(0)})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x}^{(0)})}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial f_n(\mathbf{x}^{(0)})}{\partial x_1} & \frac{\partial f_n(\mathbf{x}^{(0)})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x}^{(0)})}{\partial x_n} \end{pmatrix}.$$

Solving (3.12) for $\mathbf{x}$,

$$\mathbf{x} = \mathbf{x}^{(0)} - J(\mathbf{x}^{(0)})^{-1} F(\mathbf{x}^{(0)})$$

and the resulting method is

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - J(\mathbf{x}^{(k)})^{-1} F(\mathbf{x}^{(k)}), \qquad k = 0, 1, 2, 3, \ldots.$$

This is Newton's method for solving the nonlinear system (3.10) [5].

In actual practice, we do not invert $J(\mathbf{x}^{(k)})$. Instead we solve a linear system for a correction term to $\mathbf{x}^{(k)}$: find $\delta^{(k+1)}$ from:

$$J(\mathbf{x}^{(k)})\delta^{(k+1)} = -F(\mathbf{x}^{(k)})$$

now calculate:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta^{(k+1)}.$$

3.6 Solution of the Finite-Difference Equations

The discretization leads to six nonlinear algebraic equations for each grid-block. The set of nonlinear algebraic equations is solved using Newton's method method. The equations are written as;

$$f_{(P_g)_i}(W) = 0,$$

$$f_{(S_o)_i}(W) = 0,$$

$$f_{(S_w)_i}(W) = 0,$$

$$f_{(T)_i}(W) = 0,$$

$$f_{(X_2)_i}(W) = 0,$$

$$f_{(Y_5)_i}(W) = 0,$$

or

$$F_i(W) = \mathbf{0},$$

or

$$\mathcal{F}(W) = \mathbf{0}, \tag{3.13}$$

where

$$W = (w_1, w_2, \ldots, w_{N_{gb}}),$$

$$w_i = ((P_g)_i, (S_o)_i, (S_w)_i, (T)_i, (X_2)_i, (Y_5)_i),$$

$$F_i = (f_{(P_g)_i}, f_{(S_o)_i}, f_{(S_w)_i}, f_{(T)_i}, f_{(X_2)_i}, f_{(Y_5)_i}),$$

$$\mathcal{F} = (F_1, F_2, \ldots, F_{N_{gb}}),$$

and

$$i = 1, 2, \cdots, N_{gb}.$$

Then to solve the nonlinear equation system (3.13) Newton's method can be written as

$$J(W^k)\Delta^{k+1} = -\mathcal{F}(W^k) \tag{3.14}$$

$$W^{k+1} = W^k + \Delta^{k+1}.$$

This is applied iteratively at each time step.

The matrix $J(W^k)$ is the Jacobian matrix and takes the five block diagonal

form in the following matrix equation;

$$\begin{pmatrix} \chi & \chi & & & & & \chi & & & & & & & \\ \chi & \chi & \chi & & & & & \chi & & & & & & \\ & \cdot & \cdot & \cdot & & & & & \cdot & & & & & \\ & & & \cdot & \cdot & \cdot & & & & \cdot & & & & \\ \chi & & & & & \chi & \chi & & & & \chi & & & \\ & \chi & & & & \chi & \chi & \chi & & & & \chi & & \\ & & \cdot & & & & \cdot & \cdot & \cdot & & & & \cdot & \\ & & & \chi & & & & \chi & \chi & \chi & & & & \chi \\ & & & & \chi & & & & \chi & \chi & & & & \\ & & & & & \cdot & & & & & \cdot & \cdot & \cdot & \\ & & & & & & \cdot & & & & & \cdot & \cdot & \cdot \\ & & & & & & & \chi & & & & \chi & \chi & \chi \\ & & & & & & & & \chi & & & & \chi & \chi \end{pmatrix} \begin{pmatrix} \Delta_1^{k+1} \\ \Delta_2^{k+1} \\ \vdots \\ \vdots \\ \vdots \\ \Delta_i^{k+1} \\ \vdots \\ \vdots \\ \vdots \\ \Delta_{N_{gb}-1}^{k+1} \\ \Delta_{N_{gb}}^{k+1} \end{pmatrix} = - \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ \vdots \\ \vdots \\ F_i \\ \vdots \\ \vdots \\ \vdots \\ F_{N_{gb}-1} \\ F_{N_{gb}} \end{pmatrix},$$

where each block χ represents a matrix with partial derivatives of each equation with respect to each unknown variable i.e. if we ignore the index i

$$\chi = \begin{pmatrix} \frac{\partial f_{P_g}}{\partial P_g} & \frac{\partial f_{P_g}}{\partial S_o} & \frac{\partial f_{P_g}}{\partial S_w} & \frac{\partial f_{P_g}}{\partial T} & \frac{\partial f_{P_g}}{\partial X_2} & \frac{\partial f_{P_g}}{\partial Y_5} \\ \frac{\partial f_{S_o}}{\partial P_g} & \frac{\partial f_{S_o}}{\partial S_o} & \frac{\partial f_{S_o}}{\partial S_w} & \frac{\partial f_{S_o}}{\partial T} & \frac{\partial f_{S_o}}{\partial X_2} & \frac{\partial f_{S_o}}{\partial Y_5} \\ \frac{\partial f_{S_w}}{\partial P_g} & \frac{\partial f_{S_w}}{\partial S_o} & \frac{\partial f_{S_w}}{\partial S_w} & \frac{\partial f_{S_w}}{\partial T} & \frac{\partial f_{S_w}}{\partial X_2} & \frac{\partial f_{S_w}}{\partial Y_5} \\ \frac{\partial f_T}{\partial P_g} & \frac{\partial f_T}{\partial S_o} & \frac{\partial f_T}{\partial S_w} & \frac{\partial f_T}{\partial T} & \frac{\partial f_T}{\partial X_2} & \frac{\partial f_T}{\partial Y_5} \\ \frac{\partial f_{X_2}}{\partial P_g} & \frac{\partial f_{X_2}}{\partial S_o} & \frac{\partial f_{X_2}}{\partial S_w} & \frac{\partial f_{X_2}}{\partial T} & \frac{\partial f_{X_2}}{\partial X_2} & \frac{\partial f_{X_2}}{\partial Y_5} \\ \frac{\partial f_{Y_5}}{\partial P_g} & \frac{\partial f_{Y_5}}{\partial S_o} & \frac{\partial f_{Y_5}}{\partial S_w} & \frac{\partial f_{Y_5}}{\partial T} & \frac{\partial f_{Y_5}}{\partial X_2} & \frac{\partial f_{Y_5}}{\partial Y_5} \end{pmatrix}.$$

The partial derivatives required for the evaluation of Jacobian matrix are evaluated by a finite-difference approximation. To approximate these partial derivatives by recalculating the function with a small perturbation to each of the variables in turn. For example,

$$\frac{\partial f_{P_g}}{\partial P_g} \approx \frac{1}{\Delta P_g}\left[f_{P_g}(P_g + \Delta P_g, S_o, S_w, T, X_2, Y_5) - f_{P_g}(P_g, S_o, S_w, T, X_2, Y_5)\right].$$

Similar relations are used for each variable in each function, for each grid-block.

The Jacobian is evaluated only at the first iteration rather then at each iteration of each time step. Additionally, to ensure convergence and to prevent oscillations, a damping factor was used. It involves multiplying the right hand side of the equation (3.14) by a damping factor, $\xi^{(k)}$, at each iteration, i.e.,

$$J(W^k)\Delta^{k+1} = -\xi^{(k)}\mathcal{F}(W^k),$$

where $0 < \xi^{(1)} \leq \xi^{(2)} \leq \cdots \leq \xi^{(k)} \leq 1$.

The initial guess was chosen to be $W^0 = \mathbf{0}$ for the Newton's method and first damping factor , $\xi^{(1)}$, is specified in the data file.

At the each step of Newton's iteration, the linear system of equations $A\mathbf{x} = \mathbf{b}$ is to be solved. This can be done by using direct and iterative methods. The full details of discussion of these methods are given in Chapter 5.

3.7 The Computational Model

The equations and the procedure used to simulate the in-situ combustion process were developed and discussed in the preceding chapters.

A computer program was written in FORTRAN 77 to solve the equations involved i.e. to numerically simulate the in-situ combustion process.

The solution of the system of equations is iterated until the convergence criteria is satisfied. If convergence is not reached, for any time step, after a specified number of iterations is performed, the time step is reduced by half. Once convergence is reached, the dependent variables, and production and chemical reaction

rates are calculated at the new time level. The results obtained at the new time level are printed as output. The next time step is estimated, variables are updated and a new time step is started.

The simulation package is a complex collection of subroutines tied together by the main program, BUISCOM (Bath University ISCOM). The main program and all subroutines are discussed below.

A copy of the computer program is presented in the Appendix C.

Program Main BUISCOM

This part of the program is designed to call all the subroutines which make up the remaining sections of the program. The sequence of executing the main program is shown in a flow chart in the Appendix B.

Subroutine INPUT

All information necessary for the computer program to solve the mathematical model are read in this Subroutine. An example input data file is given in the Appendix D.

These data consist of:

- Reservoir Data;
    1. Reservoir dimensions,
    2. Number of grid-blocks,
    3. Slopes,
    4. Gravity.
- Initial Conditions;
    1. Pressure,

2. Water saturation,
3. Oil saturation,
4. Temperature,
5. Oxygen mole fraction in gas phase,
6. Heavy oil mole fraction in liquid phase,
7. Coke concentration.

- Band Heater Data;

1. Band heater heat,
2. Preheat time,
3. Band heater constant.

- Some Constants;

1. Permeability,
2. Thermal conductivity,
3. Gas constant,
4. Porosity,
5. Reference temperature,
6. Reference pressure,
7. Productivity constant,
8. Production pressure,
9. Maximum coke concentration,
10. Heat loss switch,
11. A switch for output file (OUTPUT.S or OUTPUT.L).

- Injection Data;
    1. Injection rates,
    2. Injection temperature.
- Capillary Pressure Data;
- Equilibrium K-values Data;
- Densities Data;
- Viscosities Data;
- Molecular Weight of Components;
- Relative Permeabilities Data;
- Enthalpies Data;
- Reaction Kinetic Data;
    1. Arrhenius constants,
    2. Activation energy,
    3. Heat of reaction,
    4. Stoichiometric coefficients.
- Numerical Control Data;
    1. Initial time step,
    2. Maximum time step,
    3. End time,
    4. Damped factor,
    5. Maximum number of Newton's iteration,

6. Maximum number of step,
7. Output frequency,
8. Time step change norms,
9. Convergence conditions,
10. Data for the solution of the system $A\mathbf{x} = \mathbf{b}$.

Subroutine SETUP

All terms which remain constant in the program during a computer run are calculated in this subroutine. It also initialises all the variables and the physical properties by calling the subroutine PHYPROPERTIES.

Subroutine PHYPROPERTIES

This subroutine calculates the following physical properties of the system;

- Capillary Pressures;
  1. Gas-oil capillary pressure,
  2. Oil-water capillary pressure.
- Equilibrium Ratios;
  1. K_1,
  2. K_2,
  3. K_3.
- Gas Phase Mole Fractions;
  1. Y_1,
  2. Y_2,

3. Y_3,

4. Y_4.

- Molecular Weights;

  1. M_g,

  2. M_o,

  3. M_w.

- Densities;

  1. Water density,

  2. Oil density,

  3. Gas density.

- Viscosities;

  1. Water viscosity,

  2. Oil viscosity,

  3. Gas viscosity.

- Relative Permeabilities;

  1. Water relative permeabilities,

  2. Oil relative permeabilities,

  3. Gas relative permeabilities.

- Enthalpies;

  1. Water enthalpy,

  2. Oil enthalpy,

  3. Gas enthalpy,

4. Coke enthalpy.

- Internal Energies;

1. Water internal energy,
2. Oil internal energy,
3. Gas internal energy,
4. Rock internal energy.

- Production Rates;
- Reaction Rates;

It also calls the routines EQUATIONCOKE and HEATLOSS.

Subroutine EQUATIONCOKE

This routine calculates the coke concentration.

Subroutine HEATLOSS

This routine calculates the rate of heat loss.

Subroutine UPDATE

This subroutine updates variables after each time step.

Subroutine FUNCT

This subroutine calculates values of all mass conservation and energy.

Subroutine JACOBIAN

This routine constructs the Jacobian containing primary variables.

Subroutine SOLVEAXB

This subroutine solves the the system of equations using LU decomposition or an iterative method.

It calls following subroutines according to user request.

- Subroutine SCALING
- Subroutine ILUFACT
- Subroutine SOLVEILU
- Subroutine DIRECT
- Subroutine GMRES
- Subroutine GMRESPRE
- Subroutine ORTHOMIN
- Subroutine ORTHOMINPRE
- Subroutine BICGSTAB
- Subroutine BICGSTABPRE

Subroutine SCALING

This subroutine scales the values in the coefficient matrix A so that the largest element in each row is unity. The scaled values of A are used to scale the elements in the b vector before the solving the system.

Subroutine ILUFACT

This subroutine is used to obtain the incomplete ILU decomposition as a preconditioner for the iterative methods.

Subroutine SOLVEILU

This subroutine is used to find the solution to the system after the ILU decomposition.

Subroutine DIRECT

This subroutine solves the system of linear equations using a direct method, LU decomposition.

Subroutine GMRES

This subroutine obtains the solution to the linear system by using GMRES method.

Subroutine GMRESPRE

This subroutine obtains the solution to the linear system by preconditioned GMRES method.

Subroutine ORTHOMIN

This subroutine obtains the solution to the linear system by ORTHOMIN method.

Subroutine ORTHOMINPRE

This subroutine obtains the solution to the linear system by preconditioned ORTHOMIN method.

Subroutine BICGSTAB

This subroutine obtains the solution to the linear system by using BI-CGSTAB

method.

Subroutine BICGSTABPRE

This subroutine obtains the solution to the linear system by preconditioned BI-CGSTAB method.

Subroutine CONVERGENCE

This routine tests the convergence of Newton's method.

Subroutine RESET

This subroutine resets values of the primary variables after failed convergence in Newton's iteration.

Subroutine TIMESTEP

This subroutine calculates new time step after a successful solution in Newton's method.

Subroutine OUTPUT

This subroutine calls other two subroutines OUTPUT1 and OUTPUT2.

Subroutine OUTPUT1

This subroutine generates output file OUTPUT.N and it contains some numerical information. An example OUTPUT.N file is given in the Appendix E.

Subroutine OUTPUT2

This subroutine generates one of the output files OUTPUT.S and OUTPUT.L

according to user request.

OUTPUT.S is a short file and contains some of the input data which read in subroutine INPUT. It also contains the pressure, oil saturation, water saturation, temperature, coke concentration and production rates.

OUTPUT.L is a long file and contains all the information which OUTPUT.S does. In addition it also contains reaction rates and physical properties like viscosity, density, equilibrium K- values, etc.

Chapter 4

Validation of the Model and Results

4.1 Validation of the Model

There are two ways to validate a model, either to compare with experimental results (i.e., laboratory experiments or field tests), or to compare with other models which are already laboratory tested and sometimes reservoir tested.

Complete and correct input data are very important in the simulation. Slight differences in certain data, defining the reservoir, such as permeability, porosity and reaction kinetic data, can cause the significant errors in the simulations.

Because of the absence of complete data for the laboratory and actual reservoir, the model was validated by comparing results of this simulator with results from two other simulators (Crookston *et al.* [15] and ISCOM [28], which are briefly presented in Chapter 1) using a common test case. Both models have been used successfully for laboratory and field scale studies during the past decade.

This test case was chosen because of the following factors; firstly because of its historic use as a bench mark test for model validation. Secondly because a complete bed(reservoir) and material property characterisation is available.

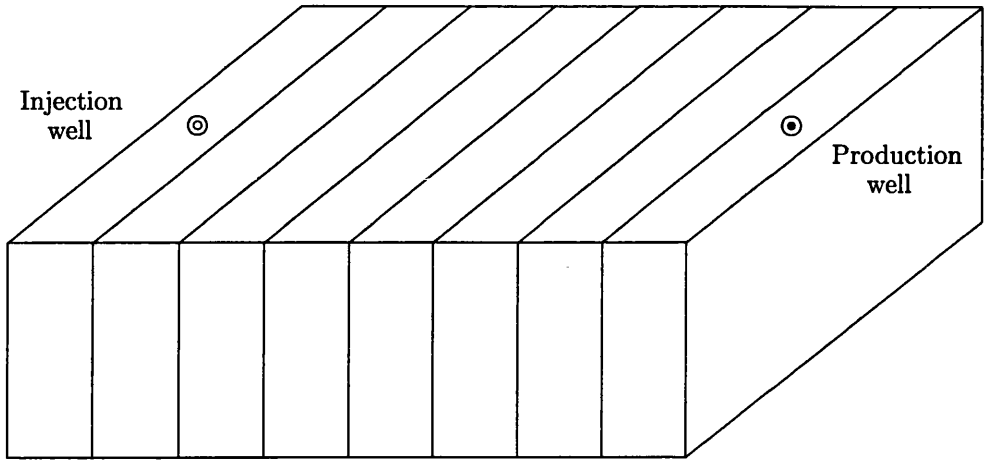


Figure 4-1: One-dimensional, 8 grid-blocks model.

The test set of reservoir description data is also used by Coats [12], Rubin and Vinsome [45] and Oklany [39].

The comparison has firstly been performed for a one dimensional dry forward combustion oil recovery process.

The test case is a reservoir 164 ft long, $115ft$ wide, and 21 ft thick. The reservoir is divided into 10 grid-blocks. Figure 4-1 illustrates a one dimensional, 8 grid-blocks model.

Initial reservoir pressure was 65.95 psi and initial reservoir temperature was 200 ^oF. Oxygen is injected into the reservoir at a rate of 300 $lb - mol/d$ at a temperature of 200 ^oF through the grid-block 1. Production was from grid-block 10. Appendix D lists the complete data employed in this simulation for validation.

The results have been compared with other two models and a fairly good match obtained. Figures 4-2, 4-3 and 4-4 show the temperature, oil saturation and water saturation profiles respectively over the reservoir for the three models, except that

because of the absence of the temperature profile of the Crookston *et al.* we made the comparison just with the ISCOM in Figure 4-2. The combustion takes place at the peak temperature point. The flat part of the temperature profile is usually called the steam plateau. From Figure 4-2 they both have almost same peak temperature at the same location of the reservoir and there is excellent agreement on the steam plateau.

In Figure 4-3 oil saturation drops to zero for the three models at the same part of the reservoir, just behind the combustion zone. Similar oil banks form in three models.

As shown in Figure 4-4, water saturation drops to zero for the three models at the same place in the reservoir. Water saturation rises above its initial value and forms a bank between the primary and secondary oil banks.

The other comparison has been made using the the results which were obtained at one of the grid-blocks: grid-block 3.

Temperature, oil saturation and water saturation curve comparisons of the present model and Crookston *et al.* [15] at grid-block 3 also illustrate good agreement as shown in Figures 4-5, 4-6 and 4-7.

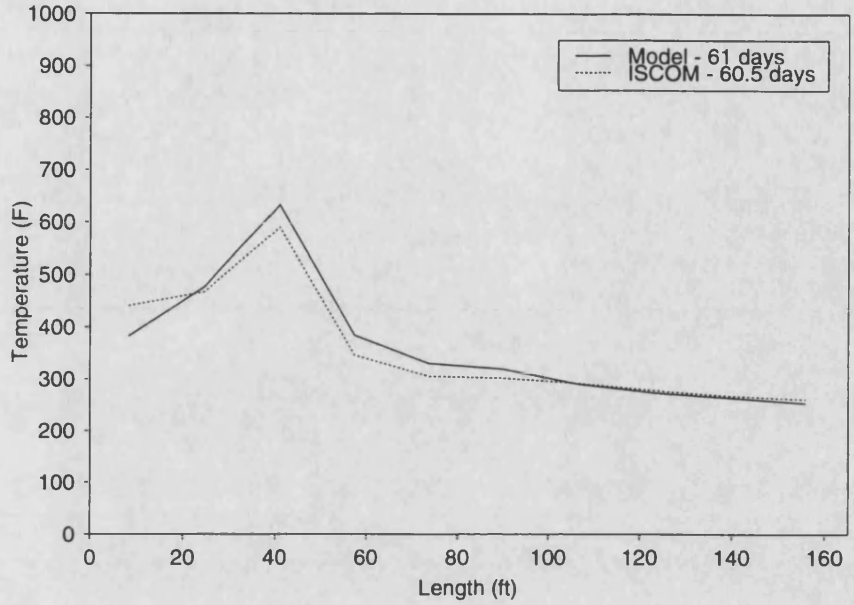


Figure 4-2: Temperature profiles over the reservoir.

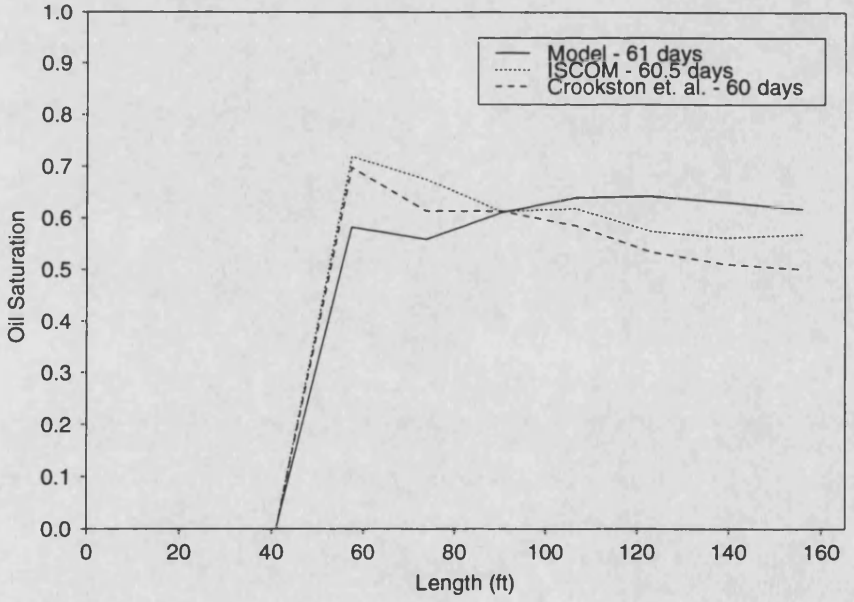


Figure 4-3: Oil saturation profiles over the reservoir.

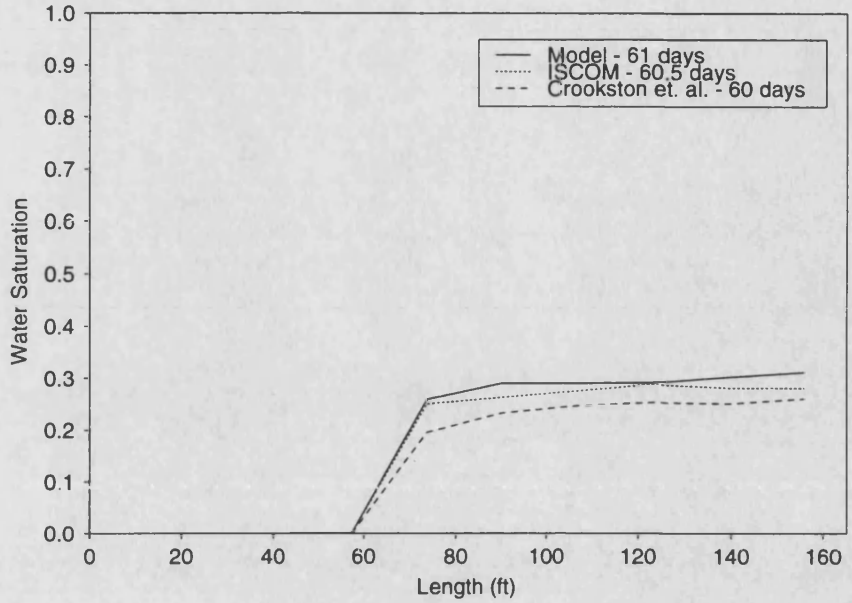


Figure 4-4: Water saturation profiles over the reservoir.

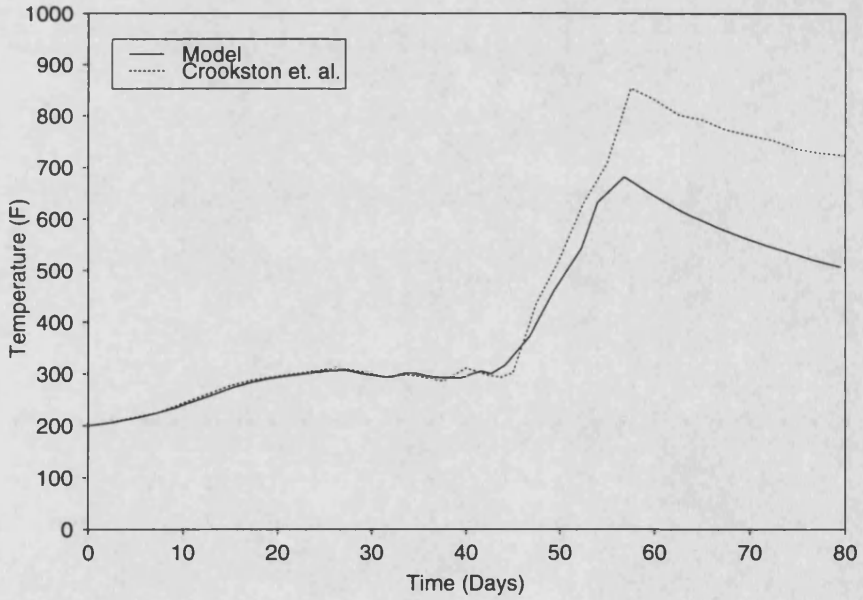


Figure 4-5: Temperature profiles of grid-block 3.

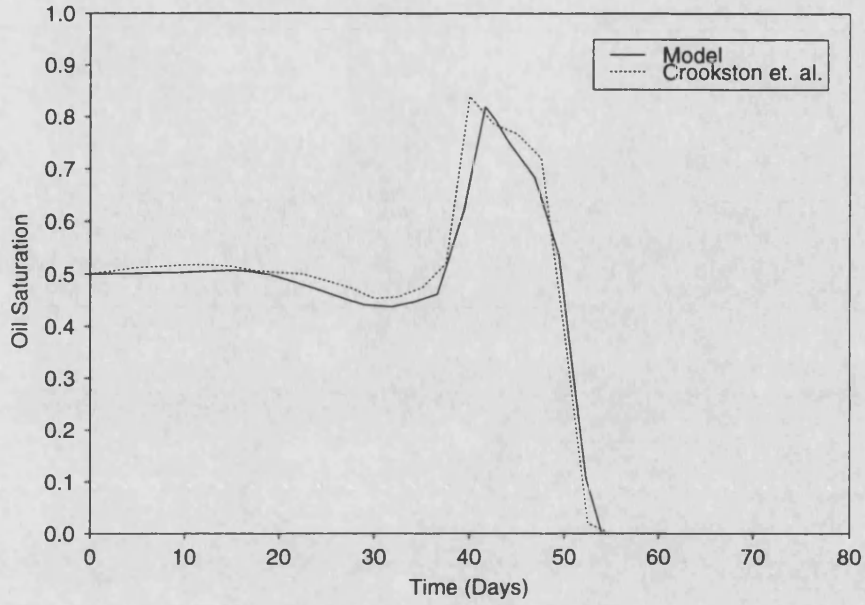


Figure 4-6: Oil saturation profiles of grid-block 3.

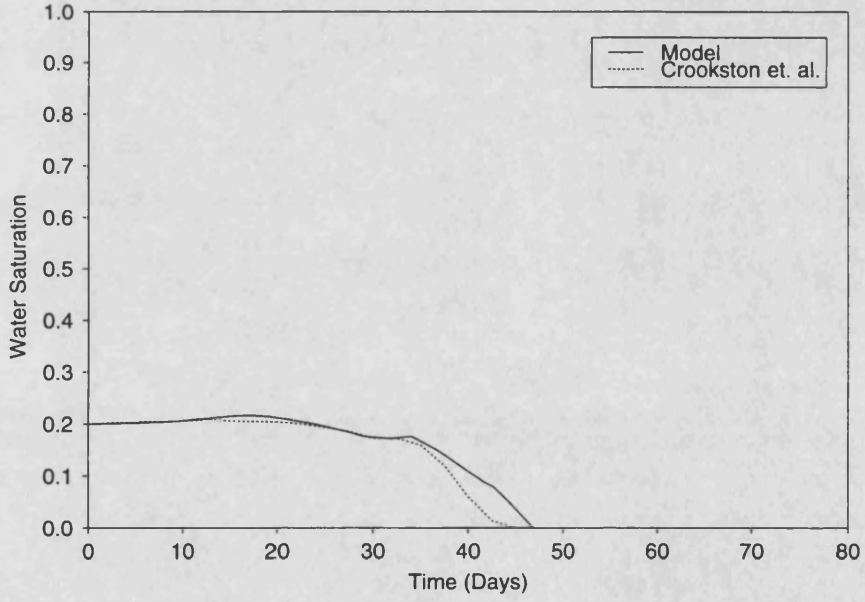


Figure 4-7: Water saturation profiles of grid-block 3.

4.2 One-Dimensional Simulation Results

To show the robustness and stability of the model further tests were carried out.

Figures 4-8, 4-9 and 4-10 present temperature, oil saturation and water saturation profiles after 25, 50, 75 and 100 days obtained from the same run.

Figure 4-8 shows the change of the reservoir temperature with distance and time. It can be noticed that combustion temperature is continuously increasing as the time increases. In all the profiles, the temperature drops ahead of the combustion zone with a sharp gradient until it reaches the steam saturation temperature, then it stabilises, forming a steam plateau. At the end of the steam plateau, the temperature drops slowly to reach the initial reservoir temperature.

The oil saturation profiles at different times are shown in Figure 4-9.

Initially, the oil saturation is the same everywhere in the reservoir and as the time progress, it increases both behind and ahead of the steam plateau, forming two peaks, as shown in Figure 4-9.

Behaviour of water saturation with time through the model is shown in Figure 4-10. Water saturation rises above its initial value and forms a bank. It coincides with the beginning of the steam plateau where the drop in temperature allows the presence of a water liquid phase. We can notice that the leading edge of the water bank drops to the level of the original water saturation.

The time variation of temperature, oil saturation and water saturation for 10 grid-blocks is shown in Figures 4-11, 4-12 and 4-13. Oil and water banking phenomena is dramatically seen in Figures 4-12 and 4-13.

The profiles all show that, at each grid-block, the same sort of "forming" and steady propagating takes place during the simulation.

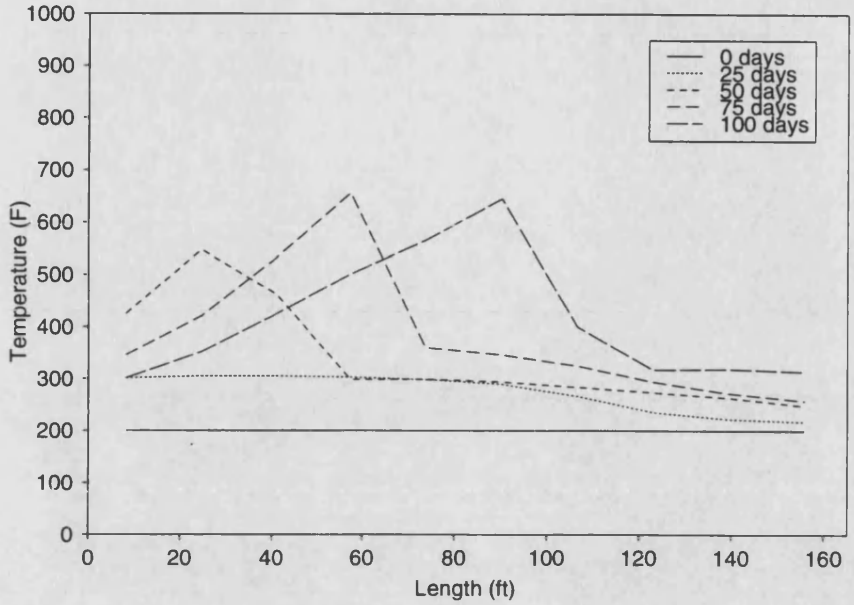


Figure 4-8: Temperature profiles over the reservoir.

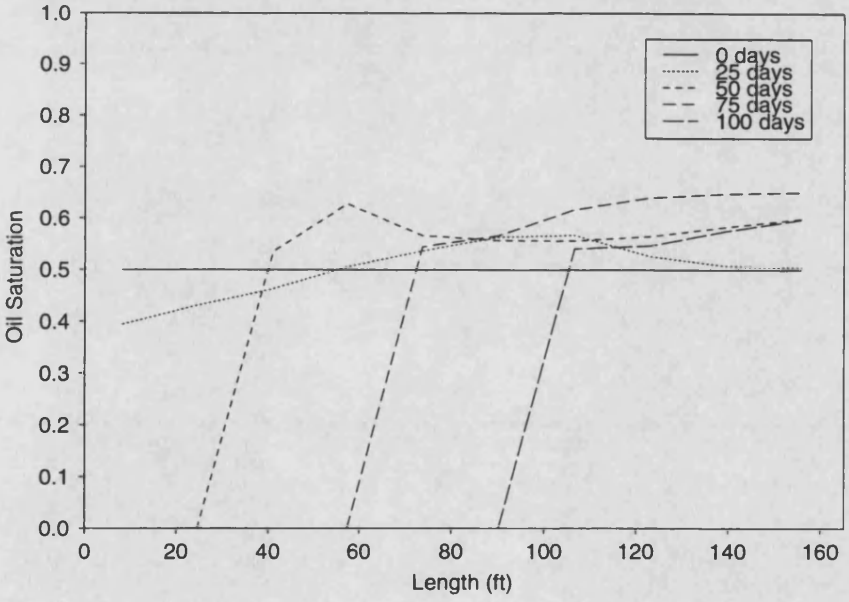


Figure 4-9: Oil saturation profiles over the reservoir.

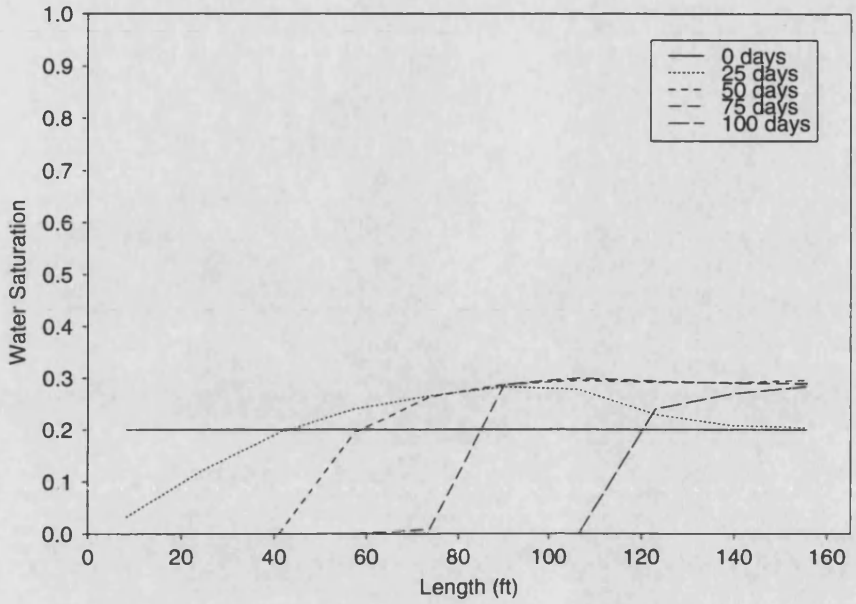


Figure 4-10: Water saturation profiles over the reservoir.

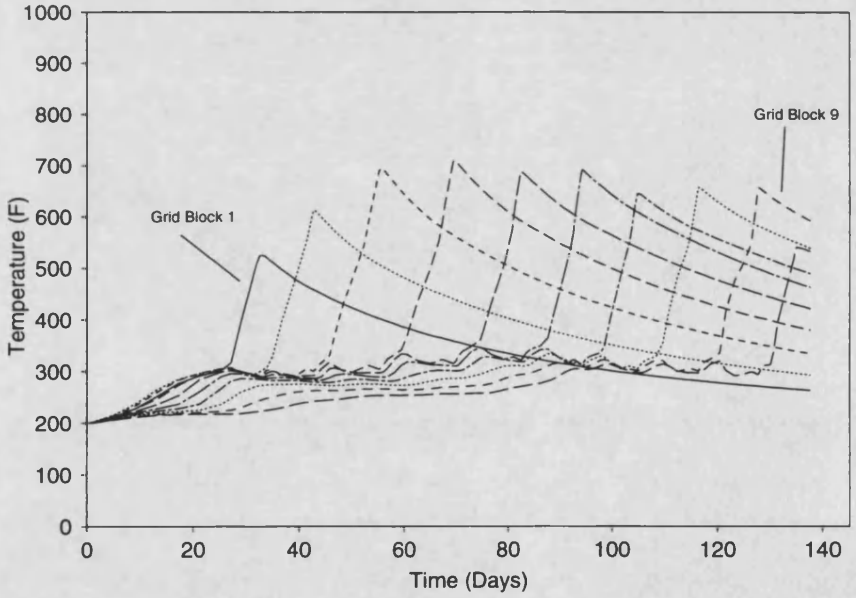


Figure 4-11: Temperature profiles of the grid-blocks.

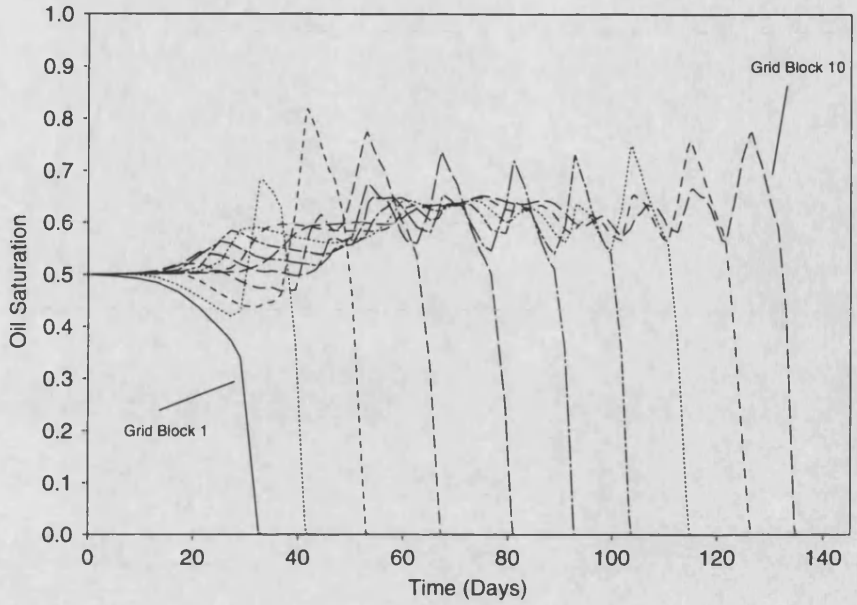


Figure 4-12: Oil saturation profiles of the grid-blocks.

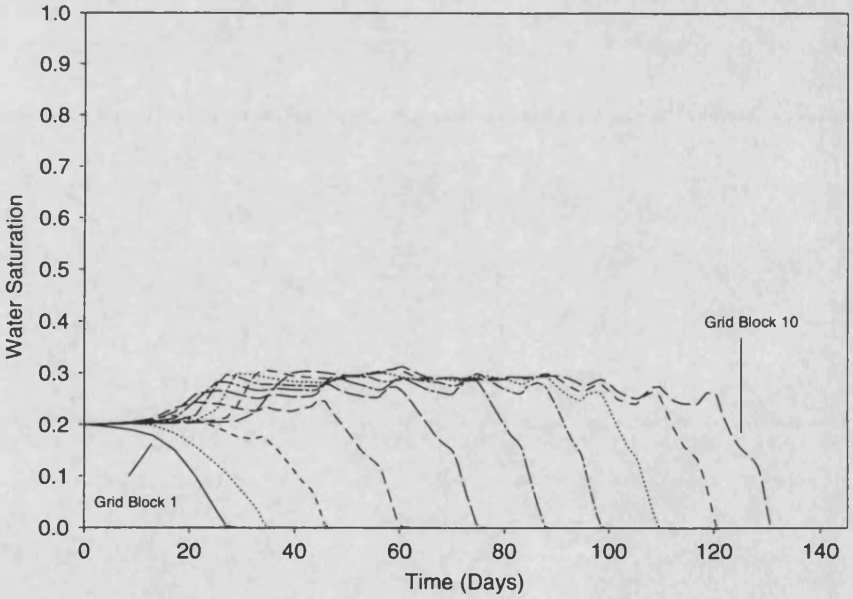


Figure 4-13: Water saturation profiles of the grid-blocks.

4.3 Two-Dimensional Simulation Results

A two-dimensional run was performed to show the combustion front movement.

A five-spot pattern geometry was assumed for the field prototype; accordingly, a symmetrical element consisting of one-quarter of the pattern, as shown in Figure 4-14, was modelled.

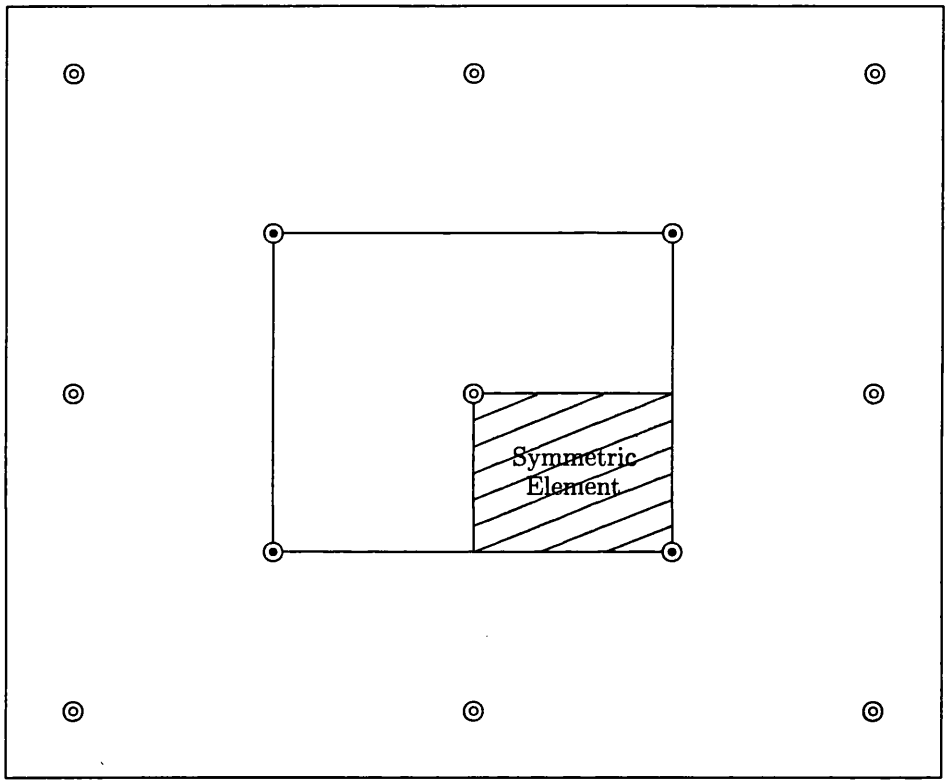


Figure 4-14: Five-spot pattern and symmetric element.
(◎ Injection well, ◉ Production well.)

In this run the example reservoir was 164 ft long, 164 ft wide, and 21 ft thick. A two-dimensional grid, 10 × 10, was used with injection and production wells at the opposite diagonal corners. Figure 4-15 illustrates a two-dimensional 8 × 8 grid blocks model.

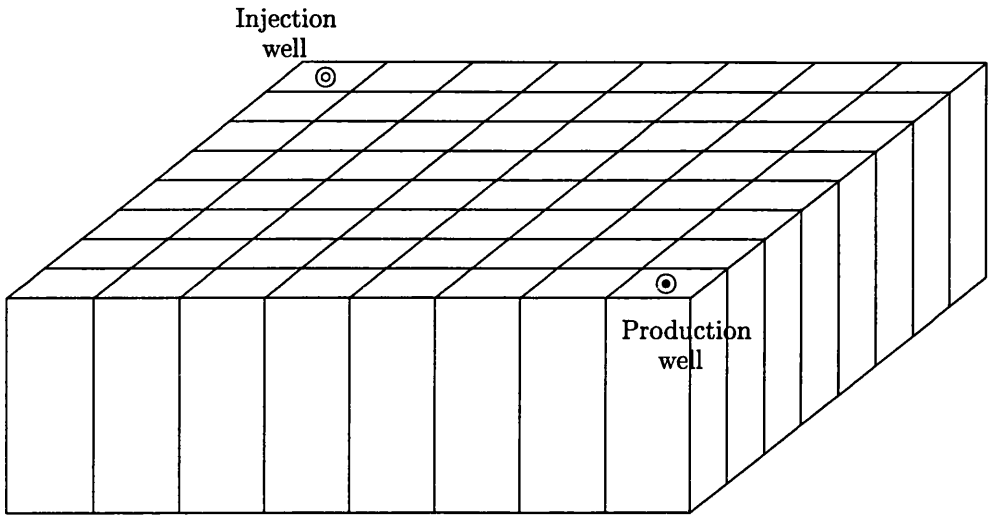


Figure 4-15: Two-dimensional, 8 × 8 grid blocks model.

The rest of the input data was the same as that used in one-dimensional run.

The detailed data can be found in Appendix D which is a copy of the input data to the present simulator.

Figures 4-16 - 4-27 show the temperature profiles and contours over the reservoir for different days. The pictures clearly indicate that the temperature front is smeared. The protrusion along the diagonal is a result of production from the production well.

The peak temperature was found to occur on the diagonal, with the temperature decreasing as one moves along the front away from the diagonal.

Figures 4-28 - 4-39 present surface and contour maps of the oil saturation obtained from the same simulation.

Each figure depicts the oil saturation distribution over the reservoir at different times.

As soon as the injection of air is initiated the oil and water banks start to build up around the injection well. Typically, the oil saturation moves with two

peaks at a constant rate. It moves immediately ahead of the combustion front.

The water bank extends further along the diagonal than along the sides, due to combustion.

Figures 4.40 - 4-51 shows that the water phase is mainly composed of the water component in the blocks surrounding the injection well. The water bank extends further along the diagonal than along the sides. This is due to the fact that there is more steam movement along the diagonal than along the grid lines. The steam then condenses, forming the water bank when it encounters a cold oil medium.

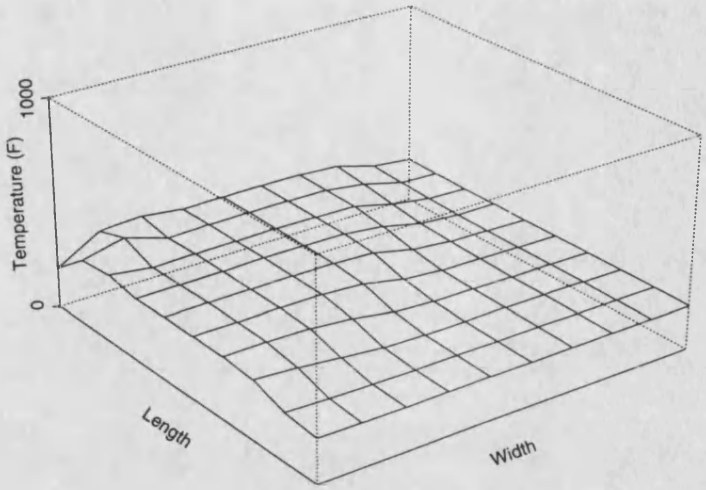


Figure 4-16: Temperature surface in 2-D reservoir at 25 days.

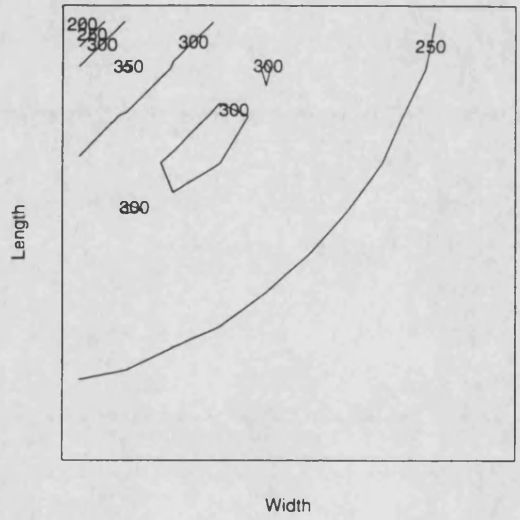


Figure 4-17: Temperature contour in 2-D reservoir at 25 days.

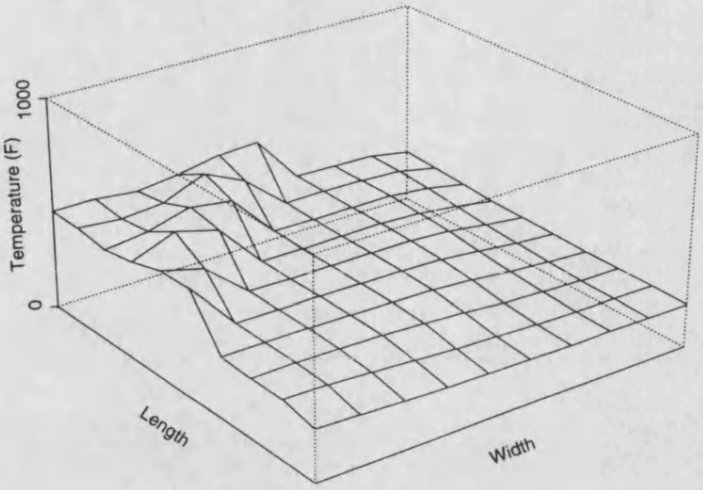


Figure 4-18: Temperature surface in 2-D reservoir at 50 days.

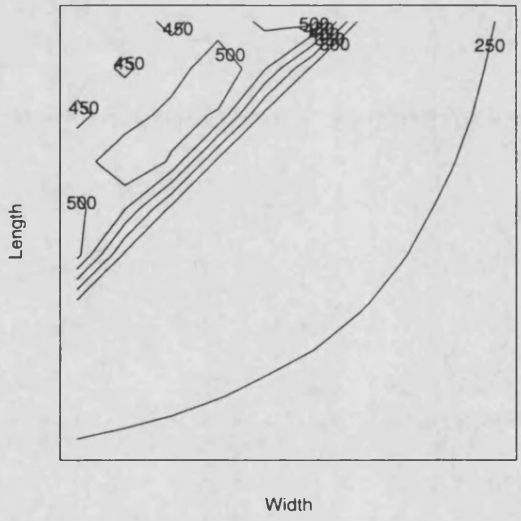


Figure 4-19: Temperature contour in 2-D reservoir at 50 days.

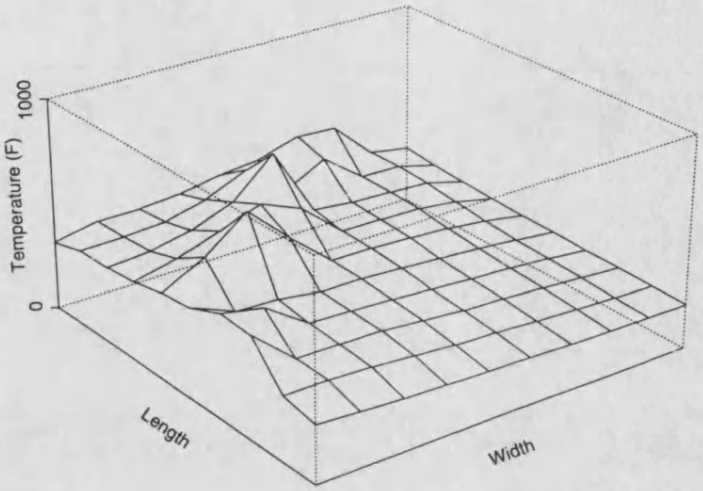


Figure 4-20: Temperature surface in 2-D reservoir at 75 days.

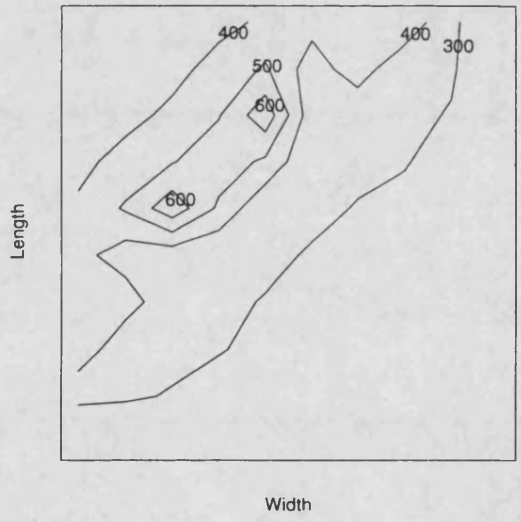


Figure 4-21: Temperature contour in 2-D reservoir at 75 days.

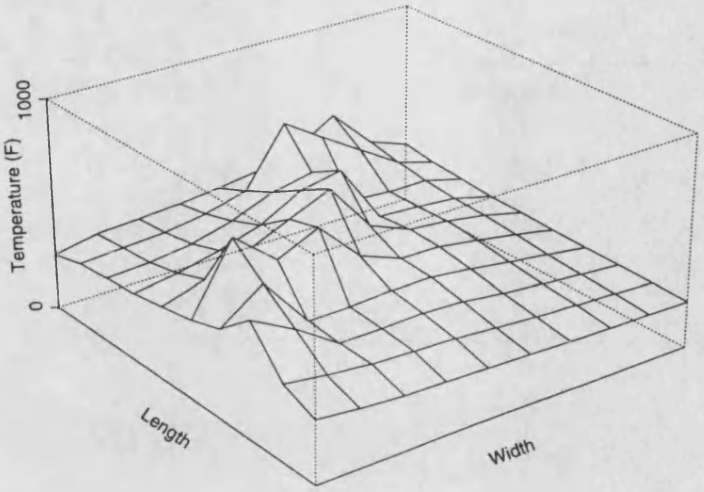


Figure 4-22: Temperature surface in 2-D reservoir at 100 days.

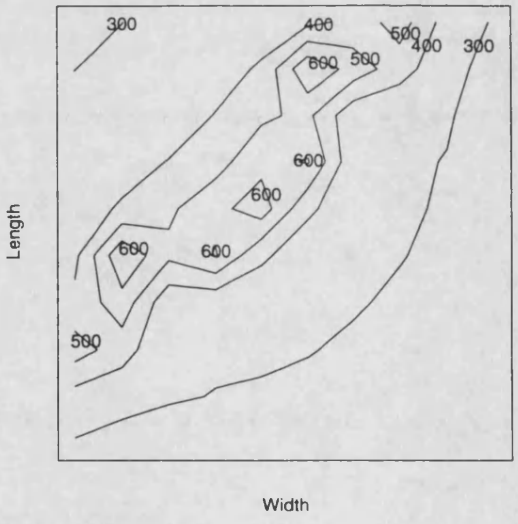


Figure 4-23: Temperature contour in 2-D reservoir at 100 days.

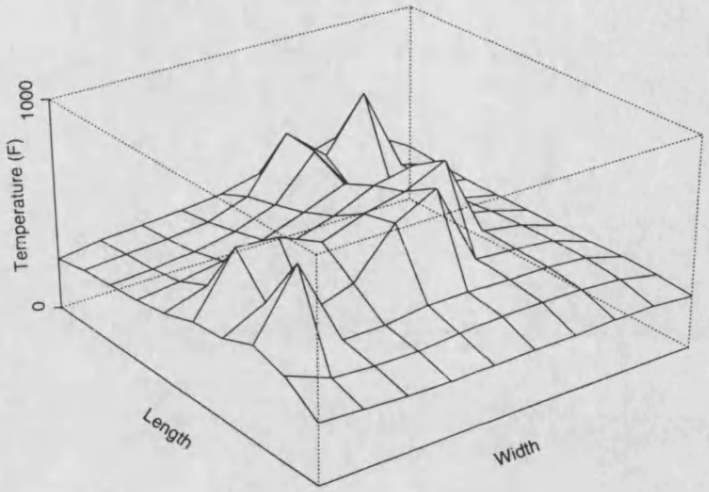


Figure 4-24: Temperature surface in 2-D reservoir at 125 days.

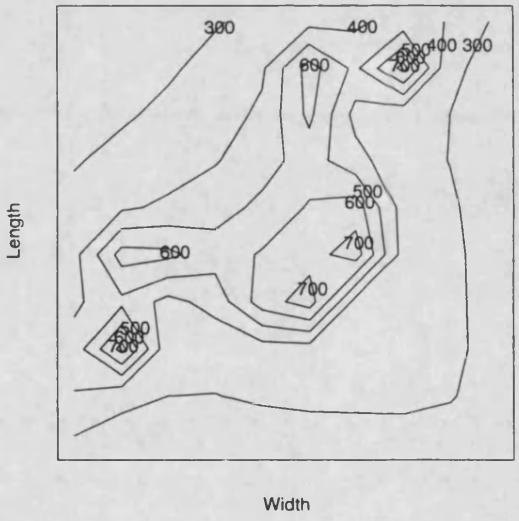


Figure 4-25: Temperature contour in 2-D reservoir at 125 days.

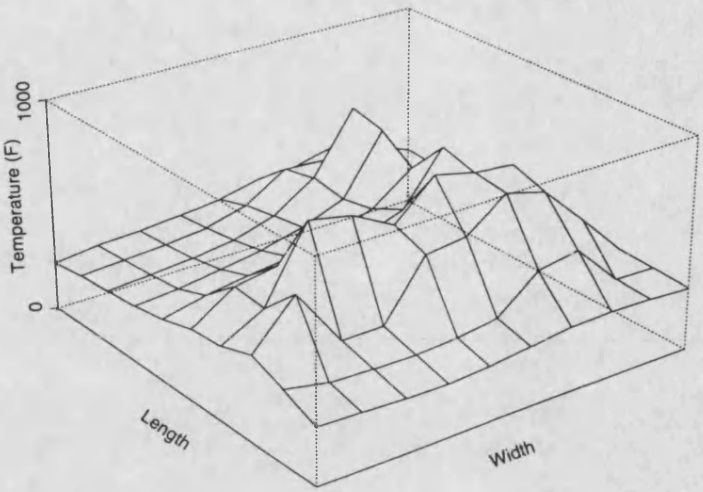


Figure 4-26: Temperature surface in 2-D reservoir at 150 days.

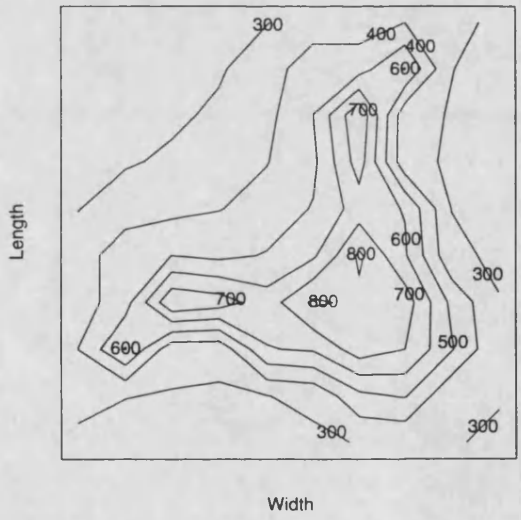


Figure 4-27: Temperature contour in 2-D reservoir at 150 days.

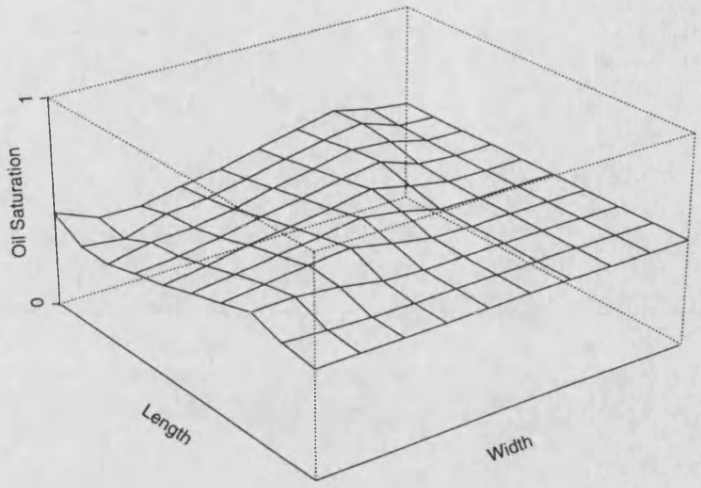


Figure 4-28: Oil saturation surface in 2-D reservoir at 25 days.

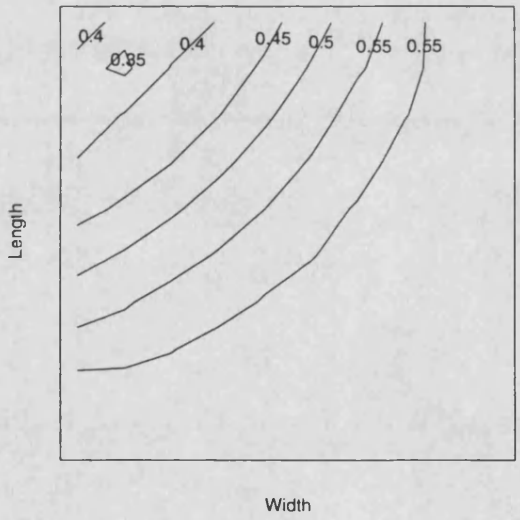


Figure 4-29: Oil saturation contour in 2-D reservoir at 25 days.

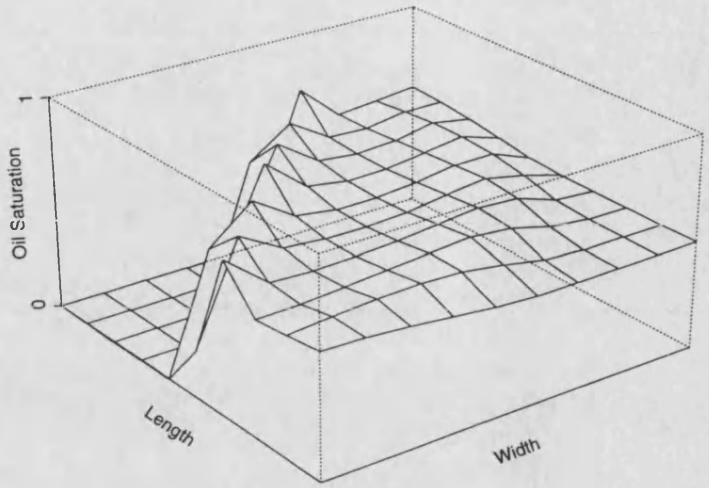


Figure 4-30: Oil saturation surface in 2-D reservoir at 50 days.

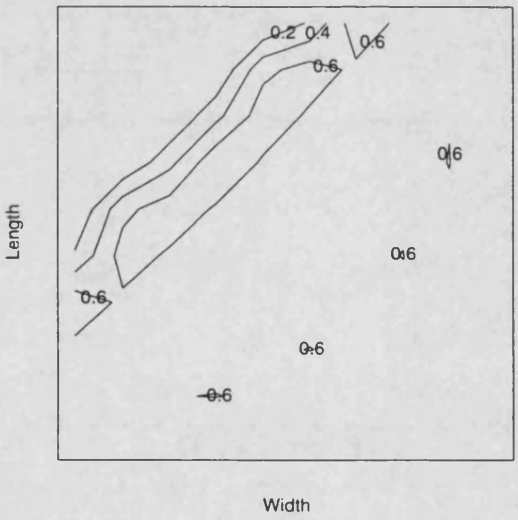


Figure 4-31: Oil saturation contour in 2-D reservoir at 50 days.

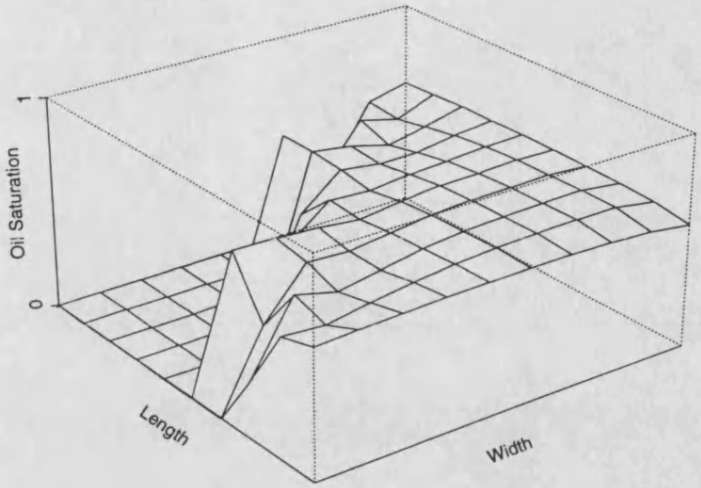


Figure 4-32: Oil saturation surface in 2-D reservoir at 75 days.

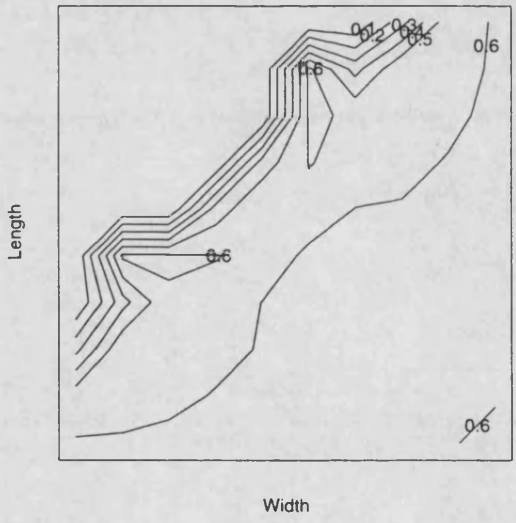


Figure 4-33: Oil saturation contour in 2-D reservoir at 75 days.

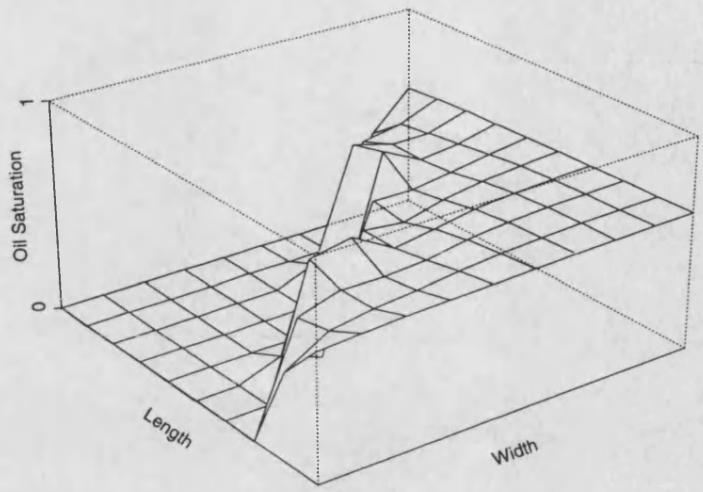


Figure 4-34: Oil saturation surface in 2-D reservoir at 100 days.

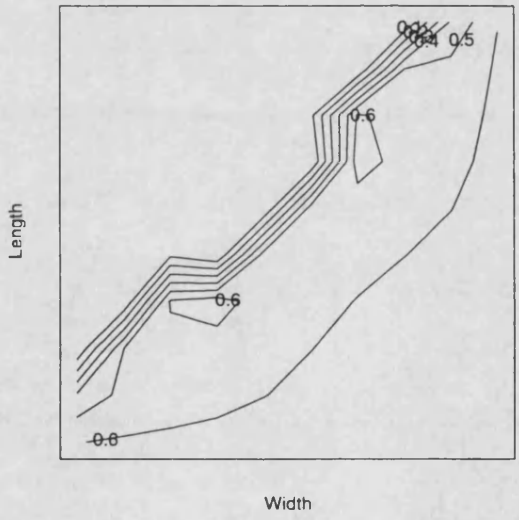


Figure 4-35: Oil saturation contour in 2-D reservoir at 100 days.

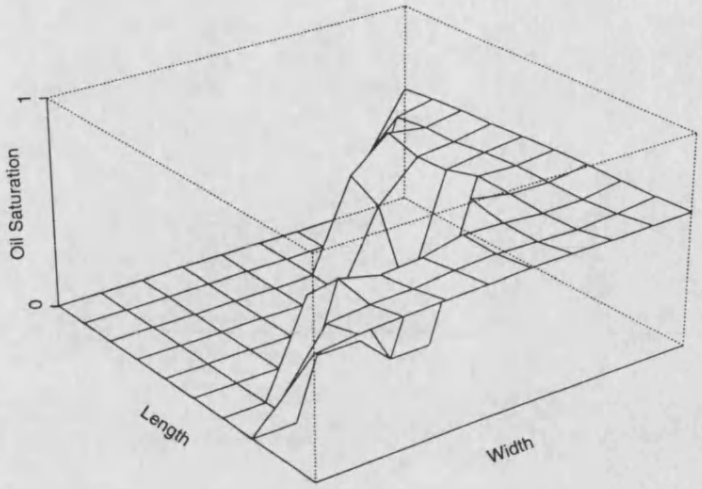


Figure 4-36: Oil saturation surface in 2-D reservoir at 125 days.

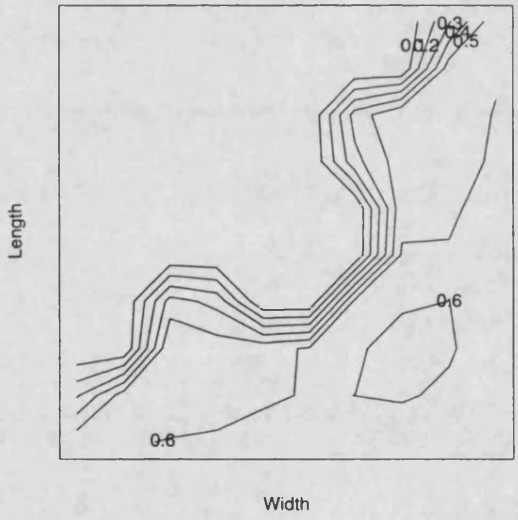


Figure 4-37: Oil saturation contour in 2-D reservoir at 125 days.

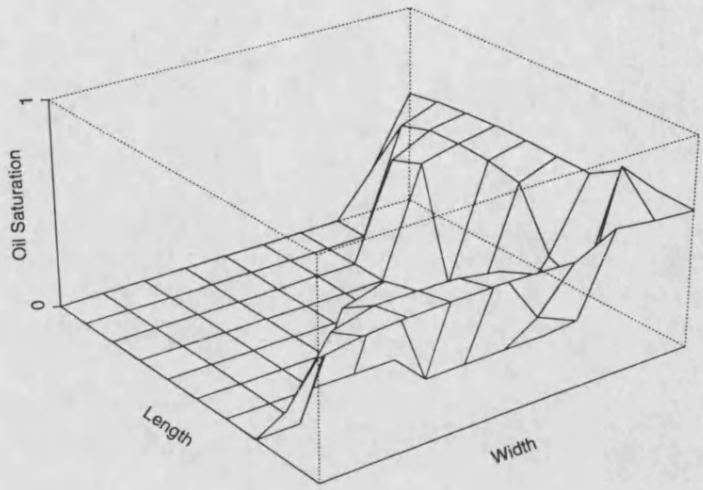


Figure 4-38: Oil saturation surface in 2-D reservoir at 150 days.

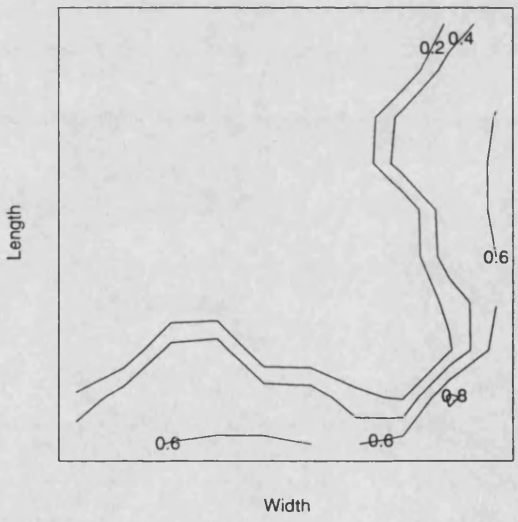


Figure 4-39: Oil saturation contour in 2-D reservoir at 150 days.

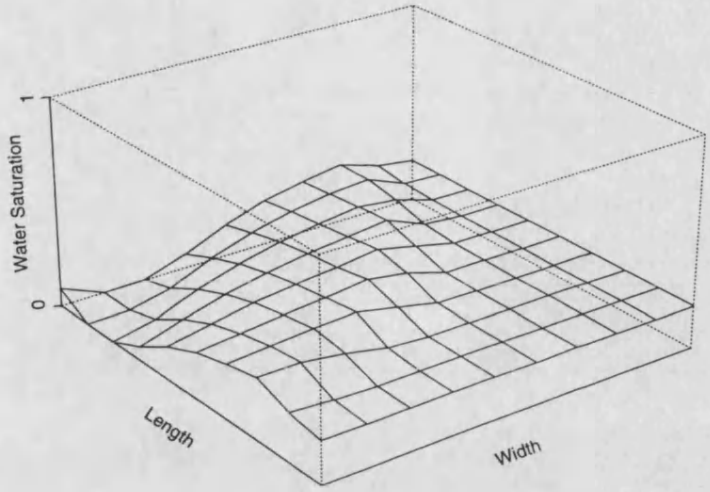


Figure 4-40: Water saturation surface in 2-D reservoir at 25 days.

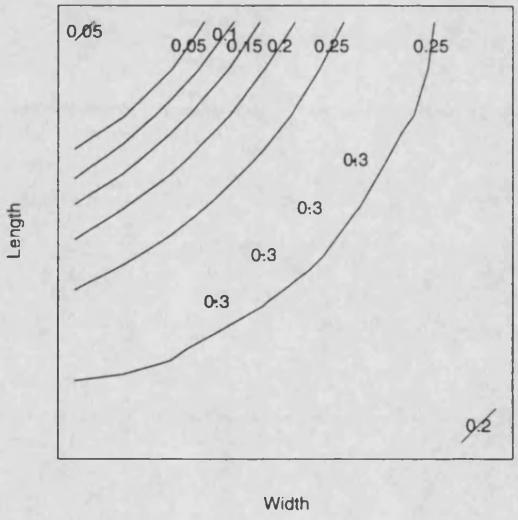


Figure 4-41: Water saturation contour in 2-D reservoir at 25 days.

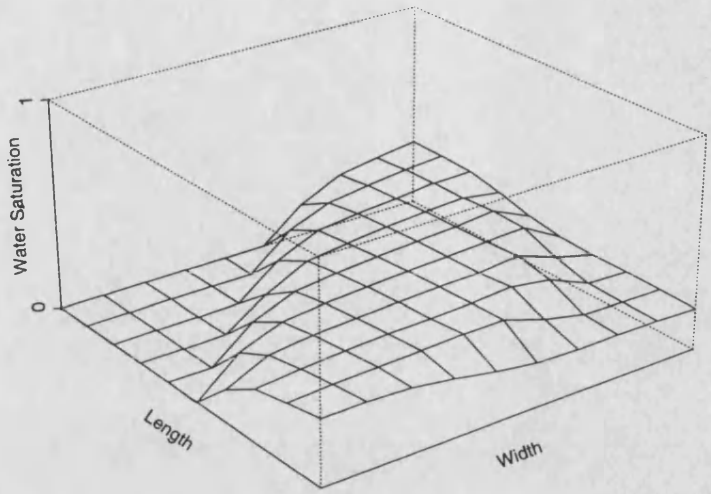


Figure 4-42: Water saturation surface in 2-D reservoir at 50 days.

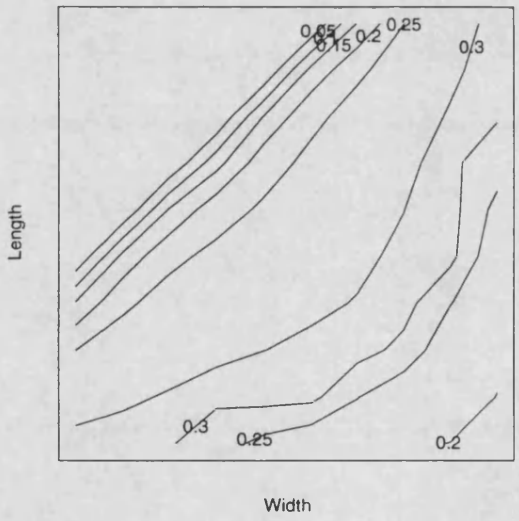


Figure 4-43: Water saturation contour in 2-D reservoir at 50 days.

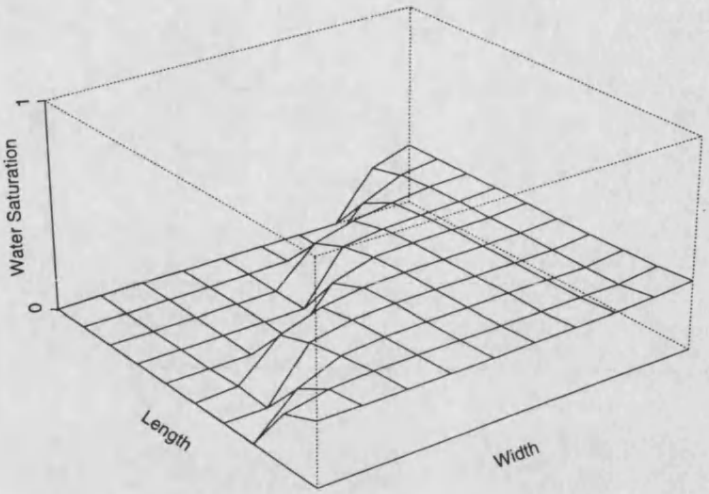


Figure 4-44: Water saturation surface in 2-D reservoir at 75 days.

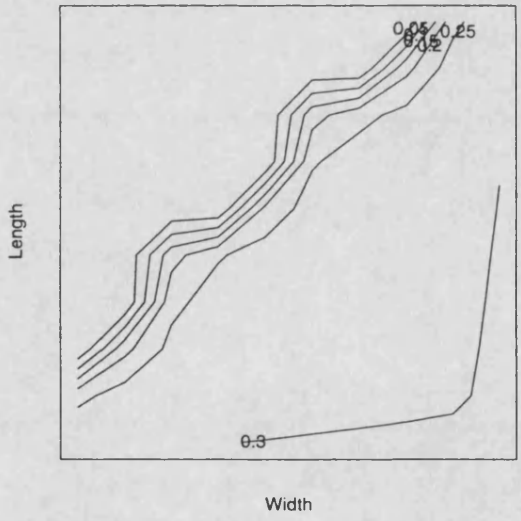


Figure 4-45: Water saturation contour in 2-D reservoir at 75 days.

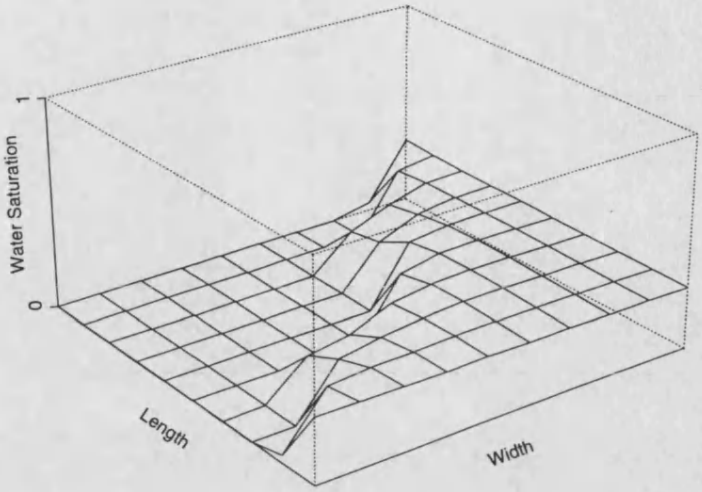


Figure 4-46: Water saturation surface in 2-D reservoir at 100 days.

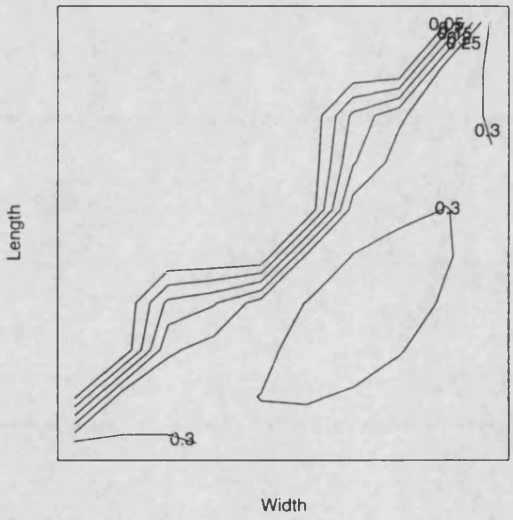


Figure 4-47: Water saturation contour in 2-D reservoir at 100 days.

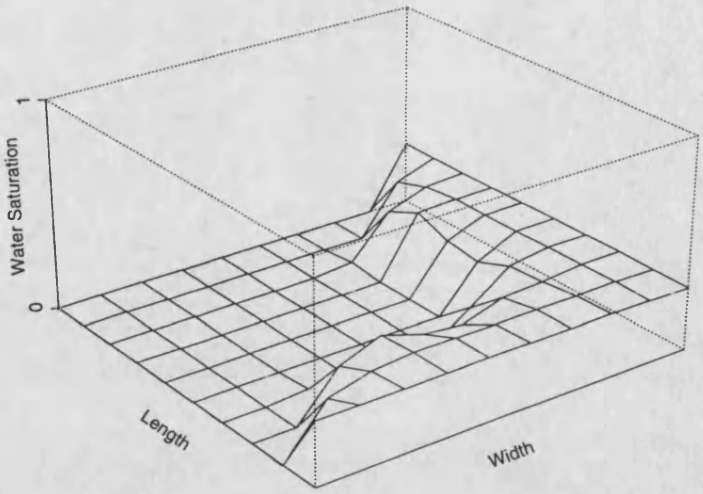


Figure 4-48: Water saturation surface in 2-D reservoir at 125 days.

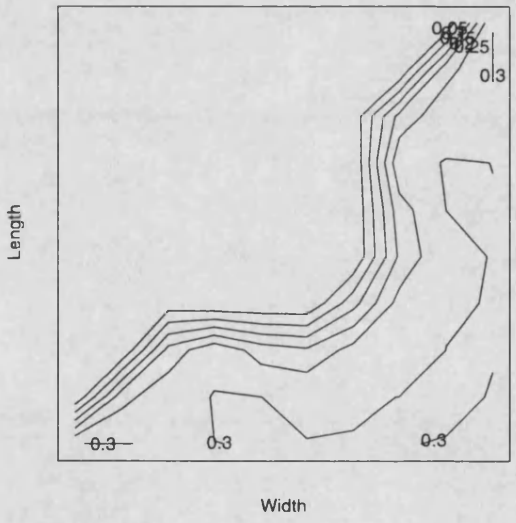


Figure 4-49: Water saturation contour in 2-D reservoir at 125 days.

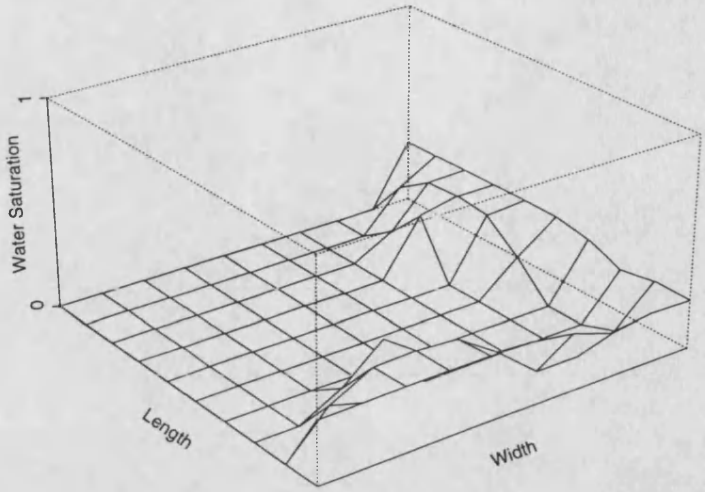


Figure 4-50: Water saturation surface in 2-D reservoir at 150 days.

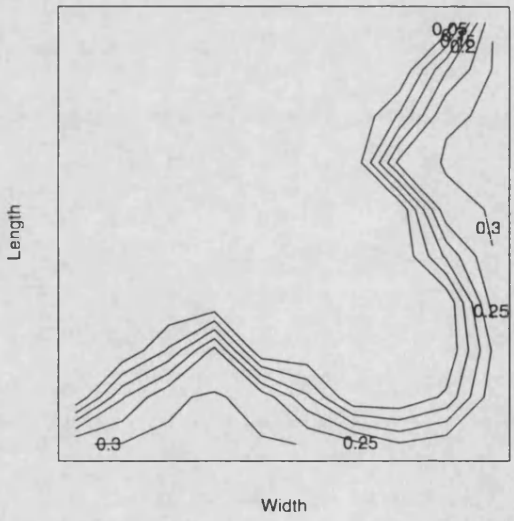


Figure 4-51: Water saturation contour in 2-D reservoir at 150 days.

4.4 One-Dimensional Grid Block Size Sensitivity Test

In order to study grid size sensitivity in the one-dimensional reservoir, several runs were made. Data used were the same as for the runs performed for the validation data.

Figures 4-52 - 4-57 show the temperature, oil saturation and water saturation profiles obtained for a number of grid-blocks; 4, 8, 16, 32 and 64 for different times, 50 and 100 days. They were used to explain the behaviour of a numerical solution under grid-block refinement.

The figures indicate that the temperature front, the head of the formed oil and water saturation tanks tend to become sharper as the number of grid-blocks changes from 4 to 8. When the number of grid-blocks double in number, 16, and 16 was doubled to 32 the same amount of changes were noticed on the profiles of the temperature, oil saturation and water saturation. However, there is a slight difference in the sharpness of the temperature front and the beginning of the "formed" oil and water banks when 32 and 64 grid-blocks were used. This can be seen in Figures 4-53, 4-55 and 4-57 clearly.

Therefore, it seems that, the use of 32 grid-blocks will probably suffice to reduce the spatial truncation errors in one dimensional reservoirs for the set of data under investigation.

An exception to this rule seems to be Figure 4-54 where convergence does not yet appear to have been achieved, even with 64 grid-blocks.

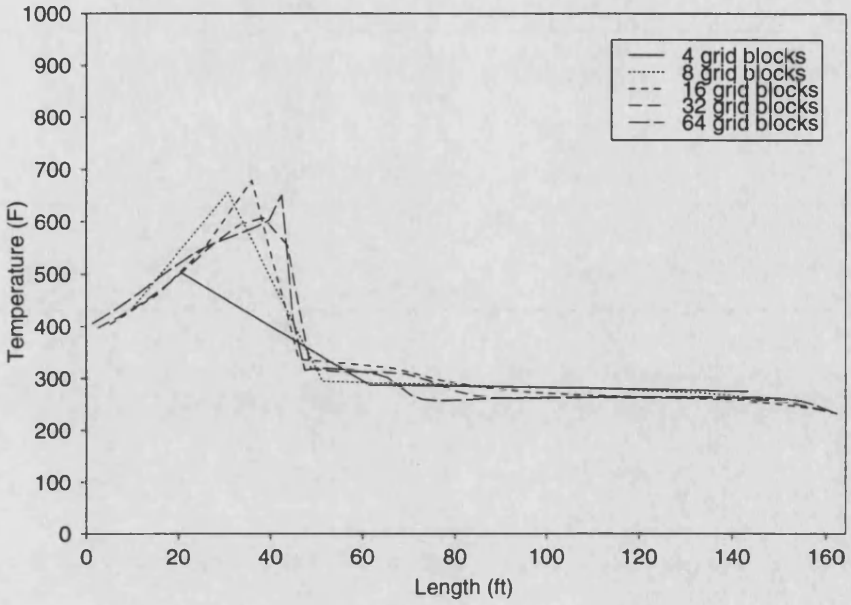


Figure 4-52: Temperature profiles over the reservoir at 50 days.

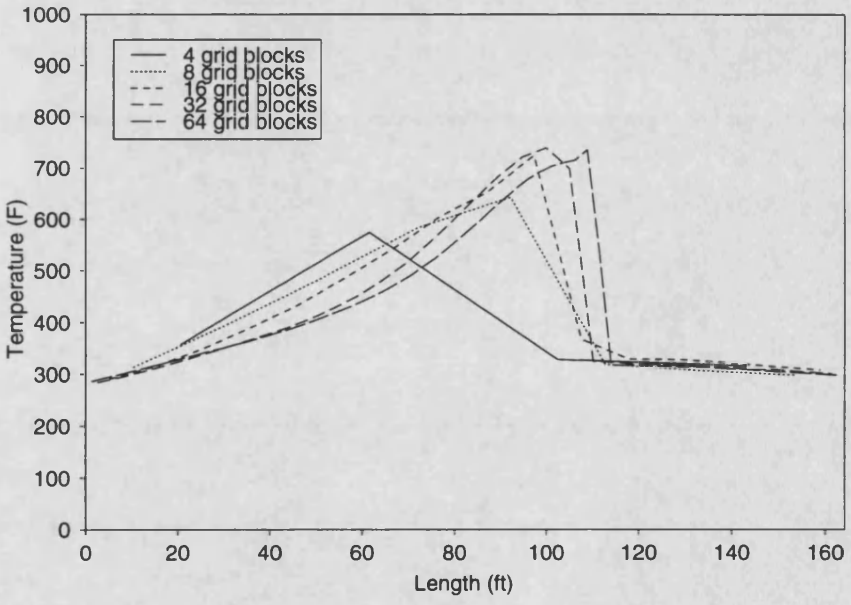


Figure 4-53: Temperature profiles over the reservoir at 100 days.

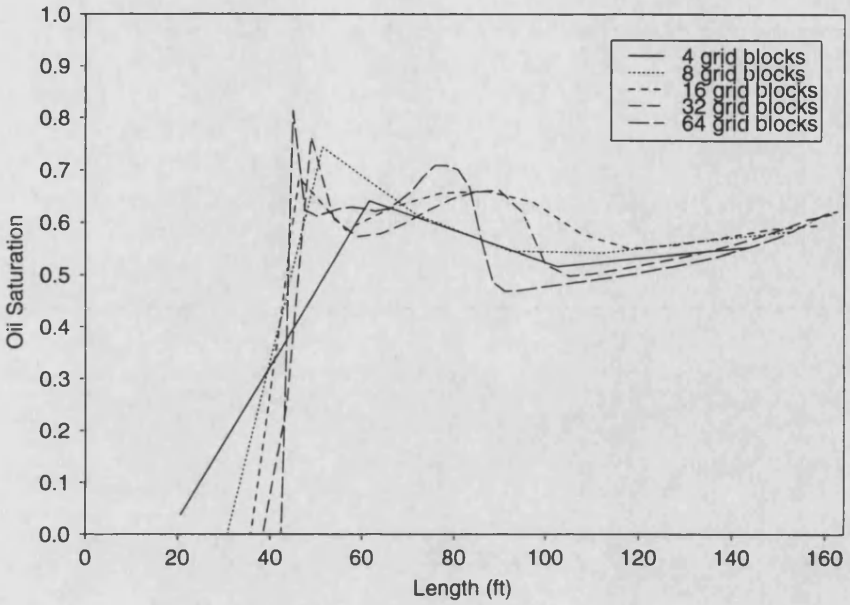


Figure 4-54: Oil saturation profiles over the reservoir at 50 days.

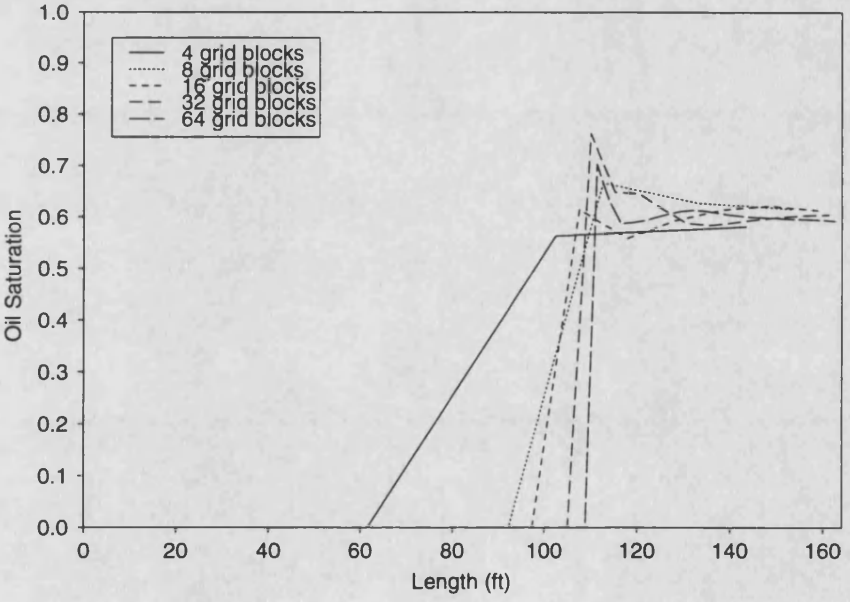


Figure 4-55: Oil saturation profiles over the reservoir at 100 days.

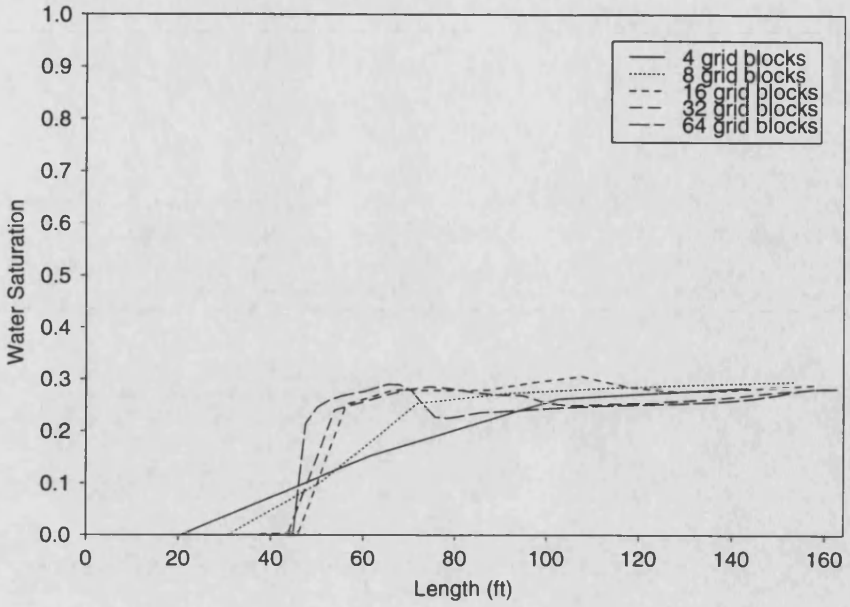


Figure 4-56: Water saturation profiles over the reservoir at 50 days.

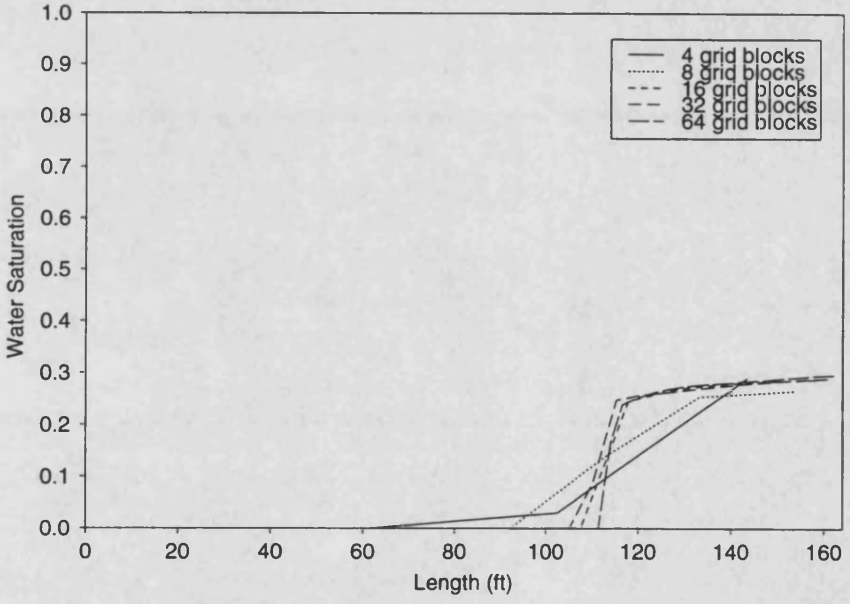


Figure 4-57: Water saturation profiles over the reservoir at 100 days.

4.5 Two-Dimensional Grid Block Size Sensitivity Test

In this section, we present the results from the runs which were made to study the sensitivity of "results" to "grid-block size" in a two-dimensional reservoir.

The same five spot element was used without any change in element reservoir volume or input data, except those necessary to accommodate changes in the number of grid-blocks.

Figures 4-58 through 4-93 show the temperature, oil saturation and the water saturation surfaces and contours using grid-blocks 4×4, 8×8 and 12×12 at two different times, namely, 75 and 150 days.

Theoretically, such an experiment should produce a sharp, vertical temperature and saturation fronts as in the one-dimensional case. It is known that, due to numerical dispersion, the finite-difference method in general smoothes out such sharp fronts.

Here again, as in the one-dimensional case, for a large number of grid-blocks, (i.e., smaller grid-block sizes), a sharper temperature front was obtained, see Figures 4-58 and 4-69.

Figures 4-70 - 4-93 show the effect of grid size on oil saturation and water saturation. The same sort of poor contour patterns were obtained from 4×4 grid-blocks tests and very similar results from the 8×8 and 12×12 grid-block cases.

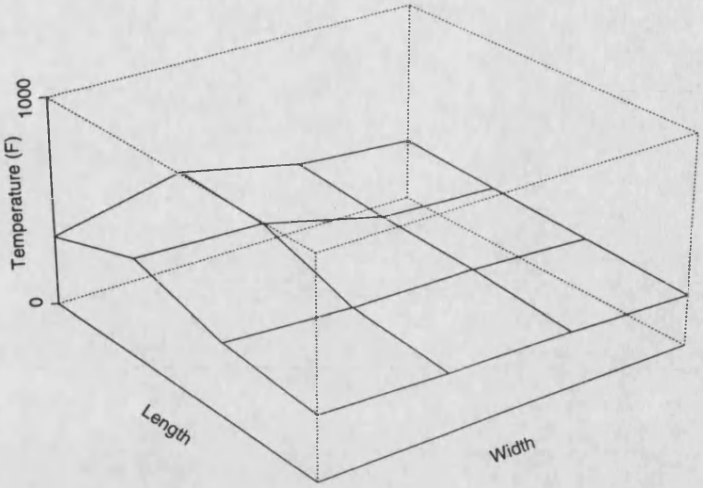


Figure 4-58: Temperature surface in 2-D reservoir with 4×4 grid-blocks at 75 days.

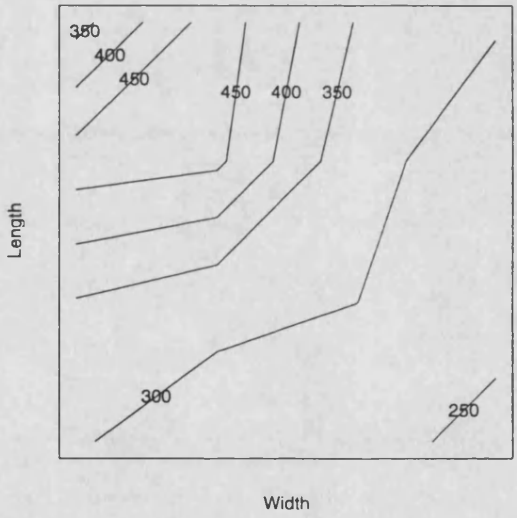


Figure 4-59: Temperature contour in 2-D reservoir with 4×4 grid-blocks at 75 days.

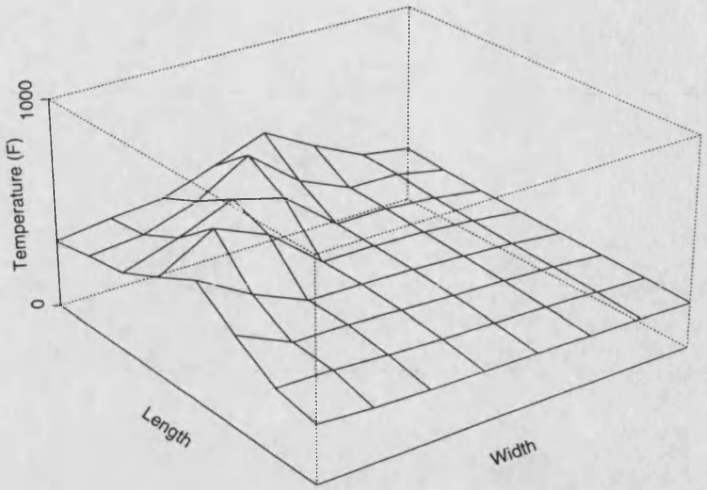


Figure 4-60: Temperature surface in 2-D reservoir with 8×8 grid-blocks at 75 days.

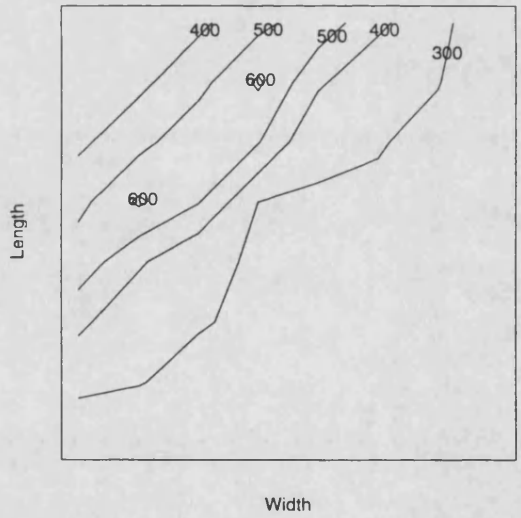


Figure 4-61: Temperature contour in 2-D reservoir with 8×8 grid-blocks at 75 days.

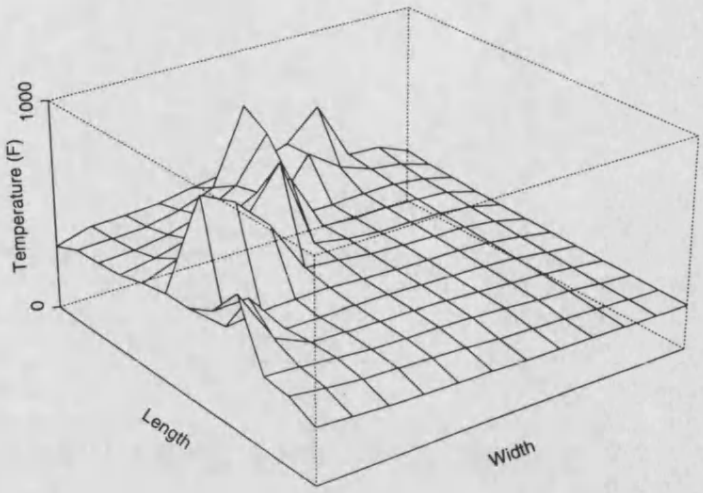


Figure 4-62: Temperature surface in 2-D reservoir with 12 × 12 grid-blocks at 75 days.

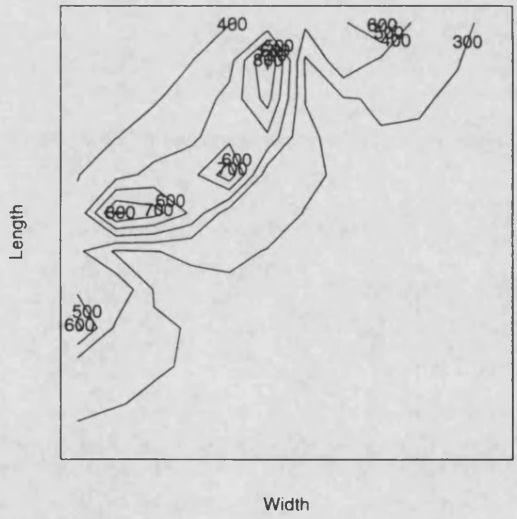


Figure 4-63: Temperature contour in 2-D reservoir with 12 × 12 grid-blocks at 75 days.

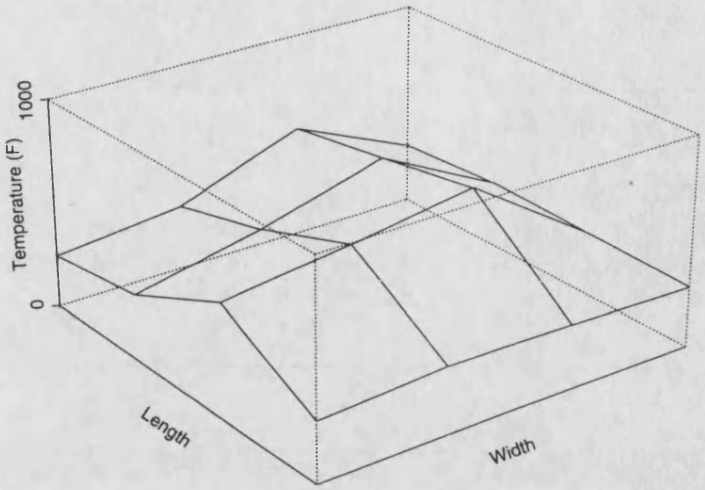


Figure 4-64: Temperature surface in 2-D reservoir with 4×4 grid-blocks at 150 days.

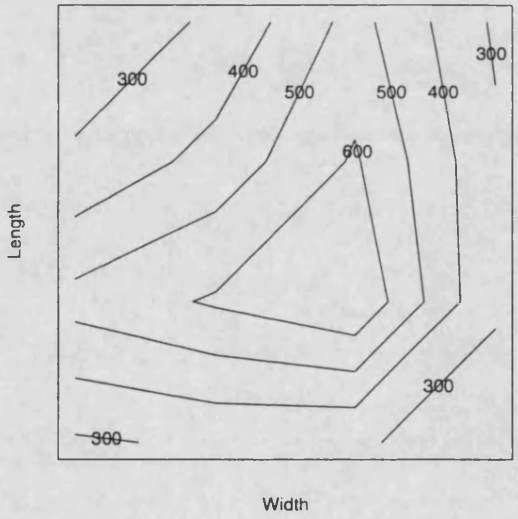


Figure 4-65: Temperature contour in 2-D reservoir with 4×4 grid-blocks at 150 days.

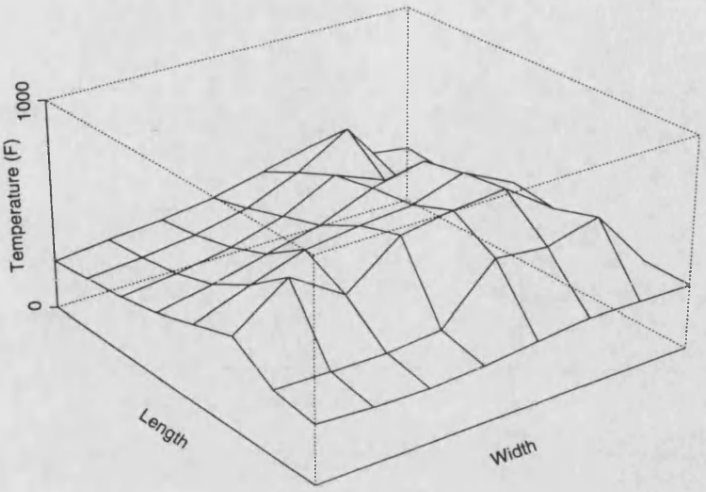


Figure 4-66: Temperature surface in 2-D reservoir with 8×8 grid-blocks at 150 days.

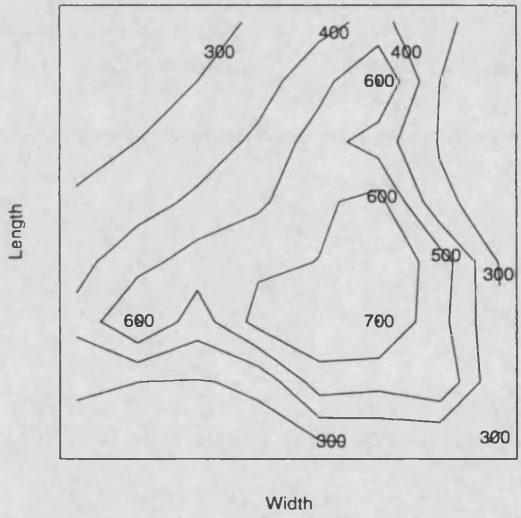


Figure 4-67: Temperature contour in 2-D reservoir with 8×8 grid-blocks at 150 days.

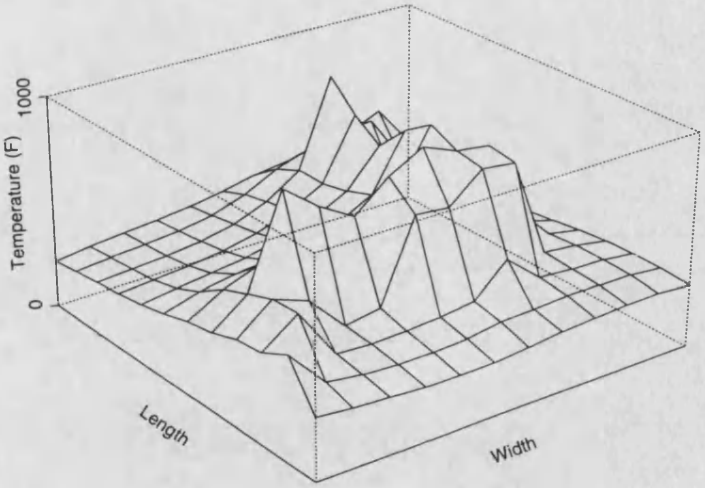


Figure 4-68: Temperature surface in 2-D reservoir with 12×12 grid-blocks at 150 days.

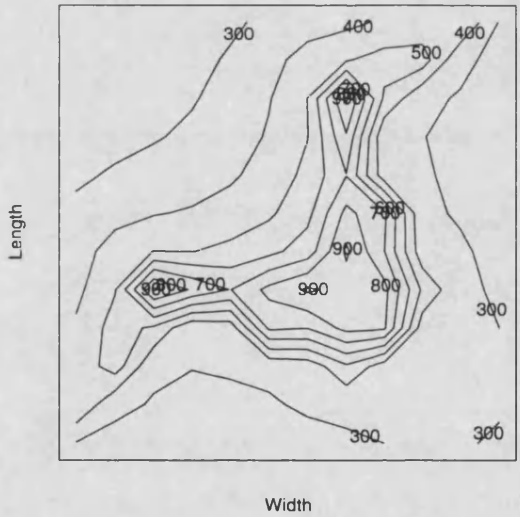


Figure 4-69: Temperature contour in 2-D reservoir with 12×12 grid-blocks at 150 days.

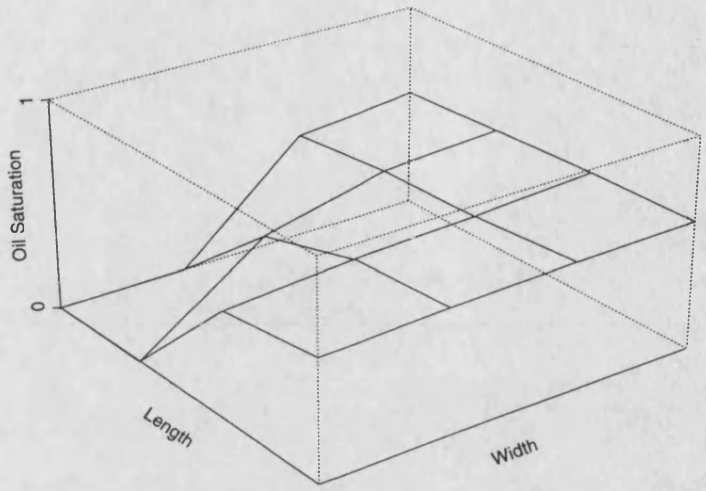


Figure 4-70: Oil saturation surface in 2-D reservoir with 4×4 grid-blocks at 75 days.

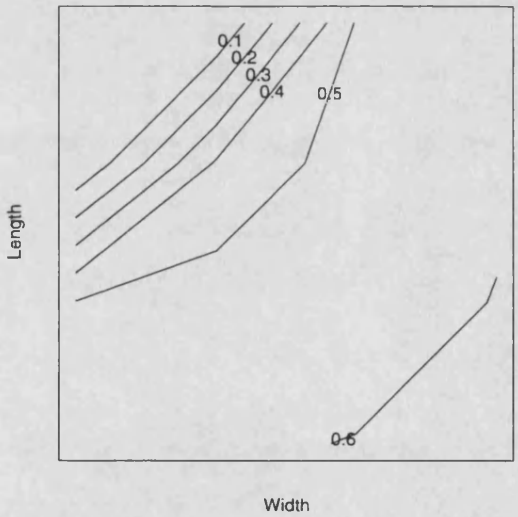


Figure 4-71: Oil saturation contour in 2-D reservoir with 4×4 grid-blocks at 75 days.

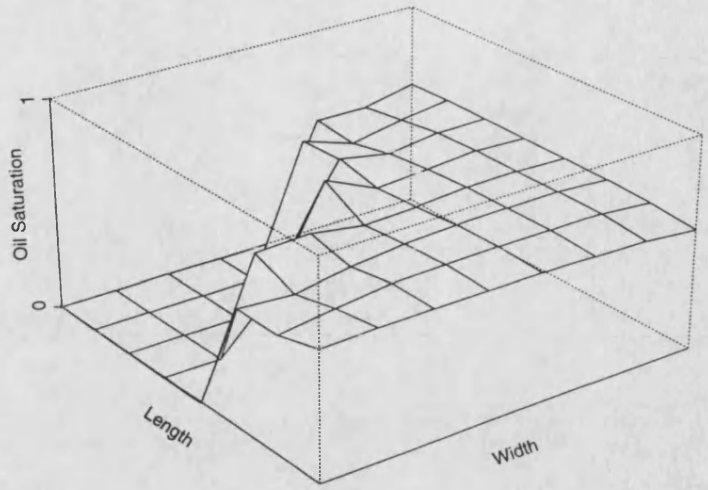


Figure 4-72: Oil saturation surface in 2-D reservoir with 8×8 grid-blocks at 75 days.

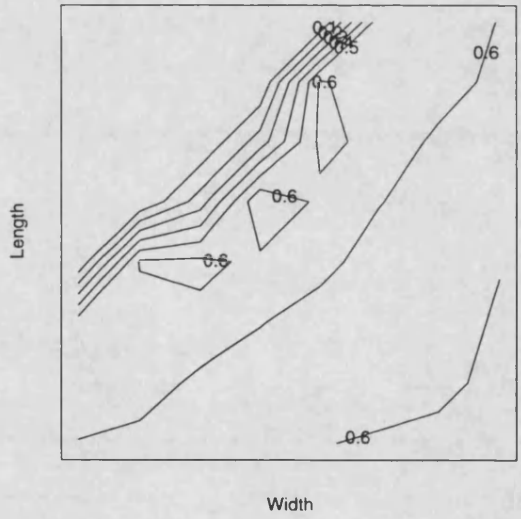


Figure 4-73: Oil saturation contour in 2-D reservoir with 8×8 grid-blocks at 75 days.

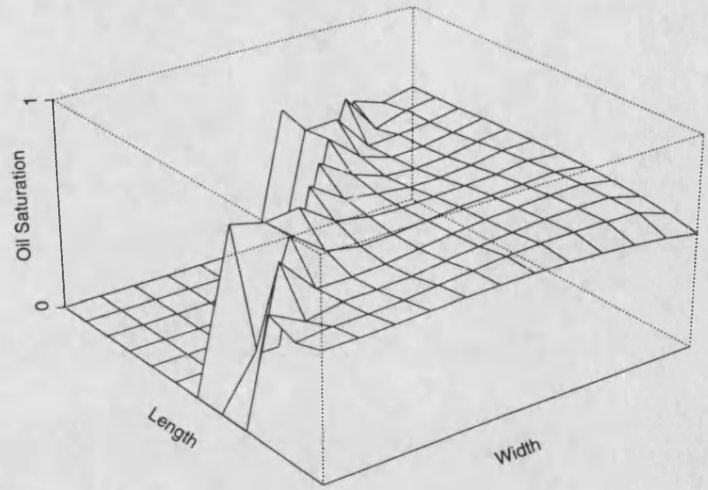


Figure 4-74: Oil saturation surface in 2-D reservoir with 12×12 grid-blocks at 75 days.

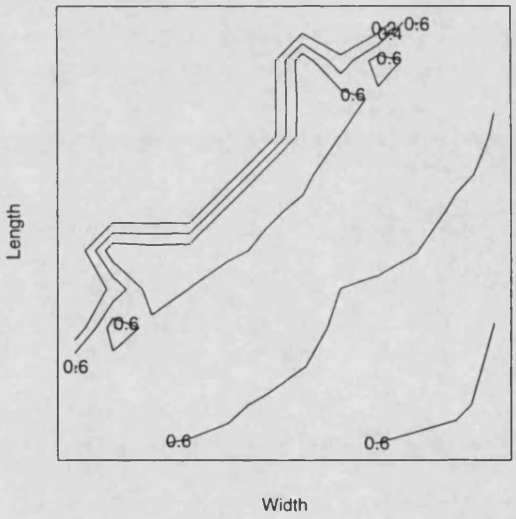


Figure 4-75: Oil saturation contour in 2-D reservoir with 12×12 grid-blocks at 75 days.

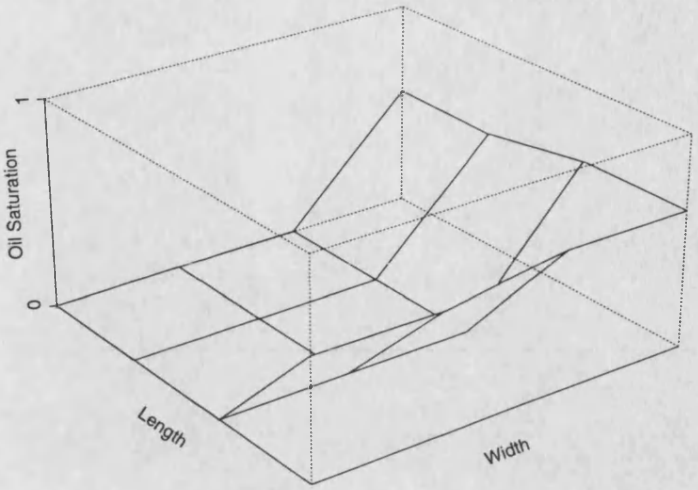


Figure 4-76: Oil saturation surface in 2-D reservoir with 4×4 grid-blocks at 150 days.

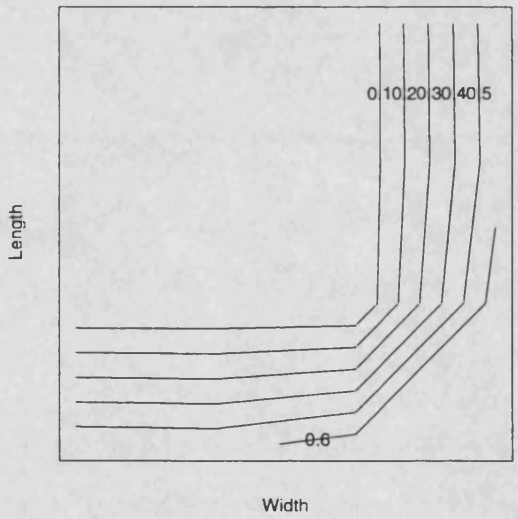


Figure 4-77: Oil saturation contour in 2-D reservoir with 4×4 grid-blocks at 150 days.

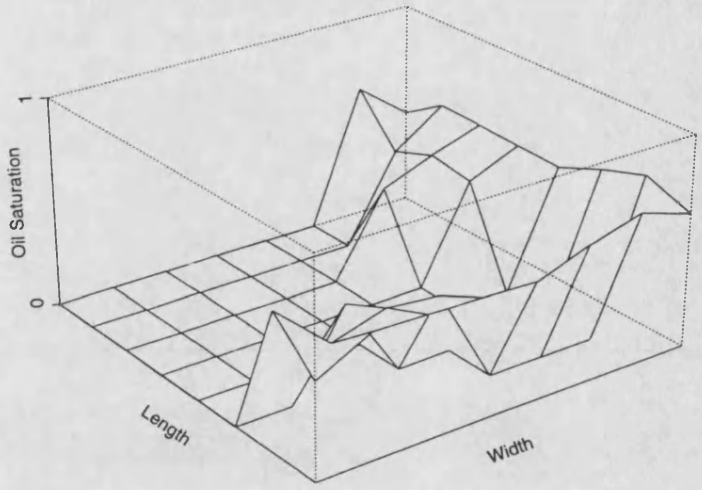


Figure 4-78: Oil saturation surface in 2-D reservoir with 8×8 grid-blocks at 150 days.

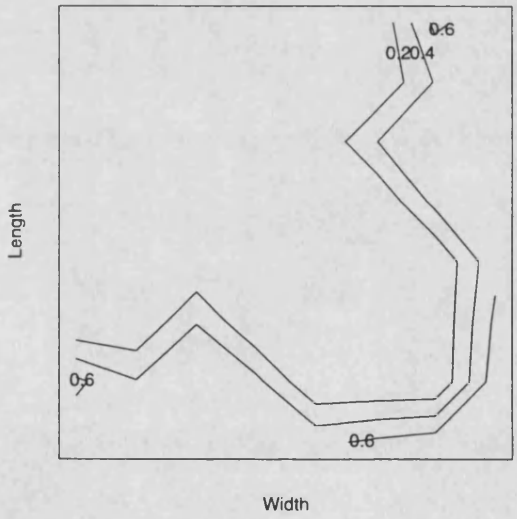


Figure 4-79: Oil saturation contour in 2-D reservoir with 8×8 grid-blocks at 150 days.

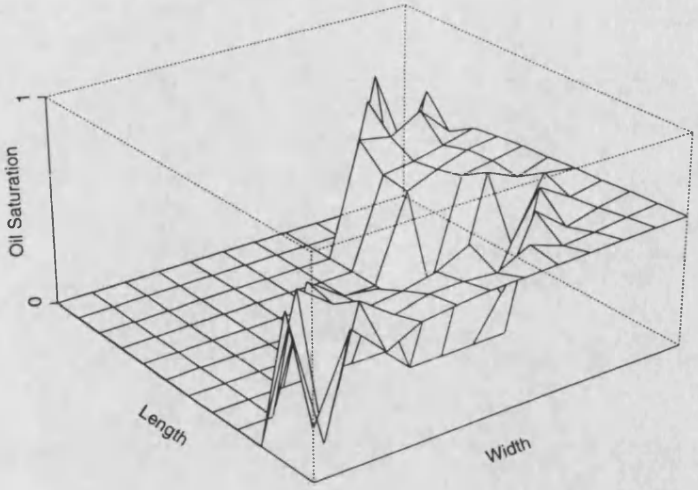


Figure 4-80: Oil saturation surface in 2-D reservoir with 12×12 grid-blocks at 150 days.

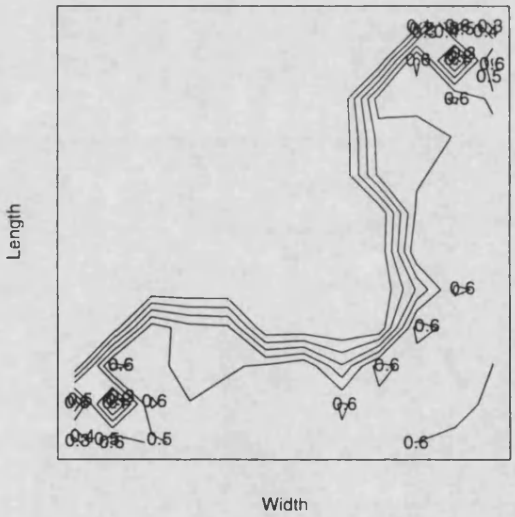


Figure 4-81: Oil saturation contour in 2-D reservoir with 12×12 grid-blocks at 150 days.

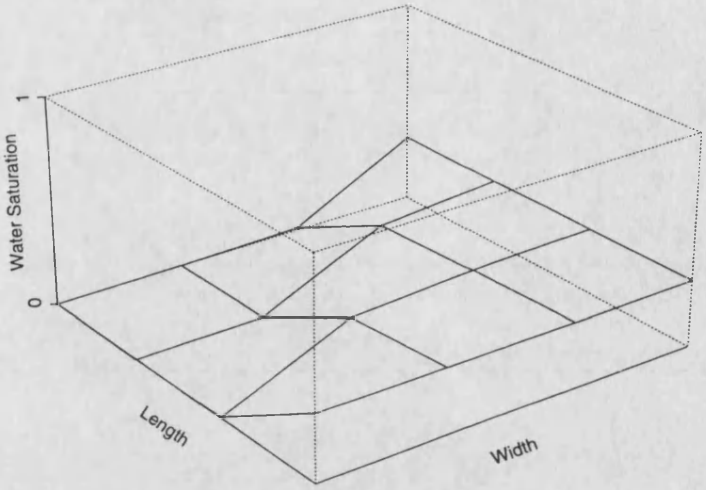


Figure 4-82: Water saturation surface in 2-D reservoir with 4×4 grid-blocks at 75 days.

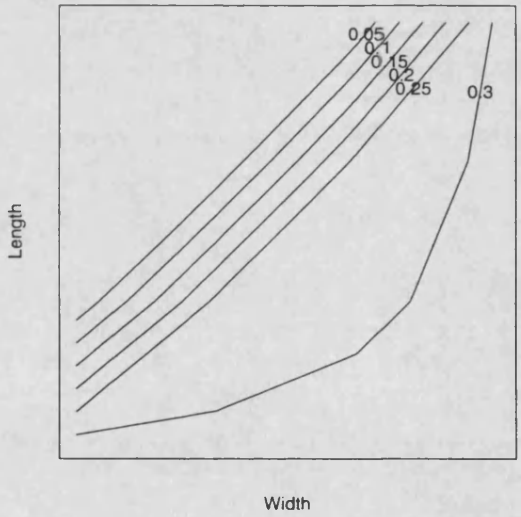


Figure 4-83: Water saturation contour in 2-D reservoir with 4×4 grid-blocks at 75 days.

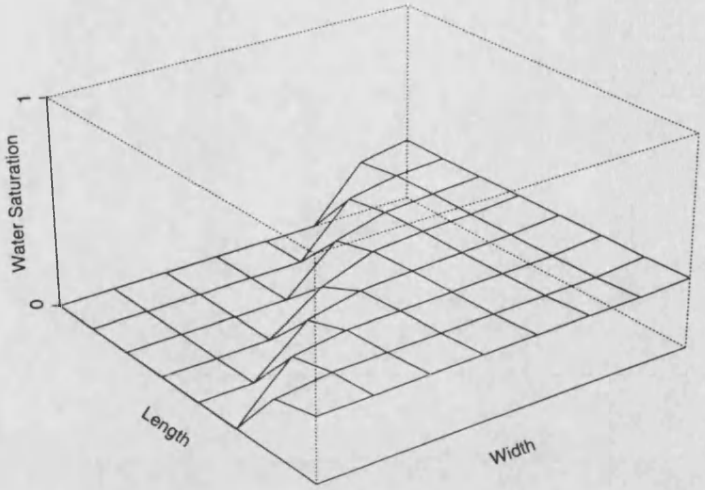


Figure 4-84: Water saturation surface in 2-D reservoir with 8×8 grid-blocks at 75 days.

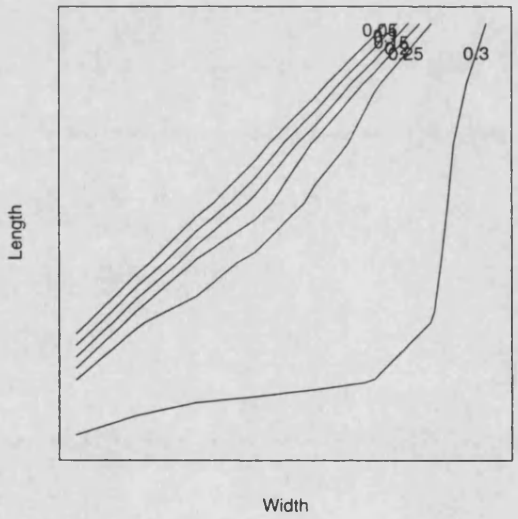


Figure 4-85: Water saturation contour in 2-D reservoir with 8×8 grid-blocks at 75 days.

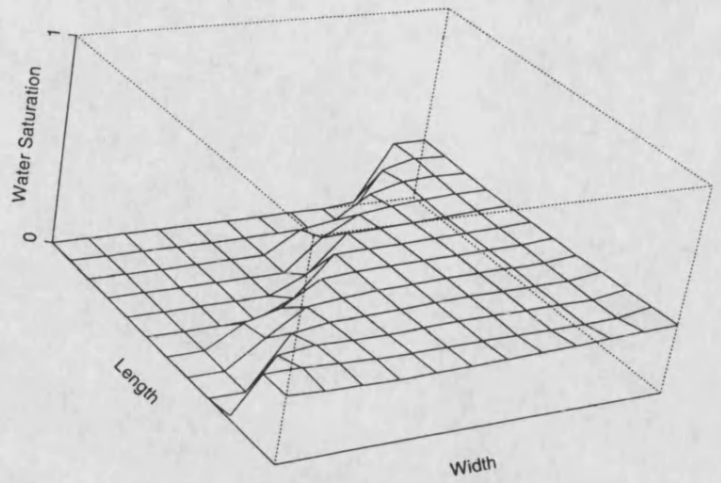


Figure 4-86: Water saturation surface in 2-D reservoir with 12×12 grid-blocks at 75 days.

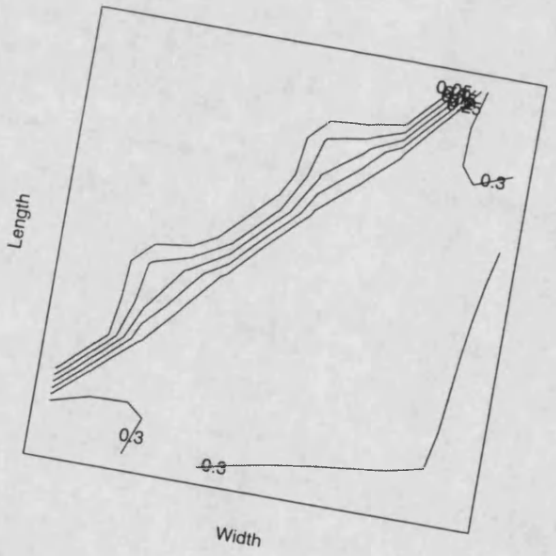


Figure 4-87: Water saturation contour in 2-D reservoir with 12×12 grid-blocks at 75 days.

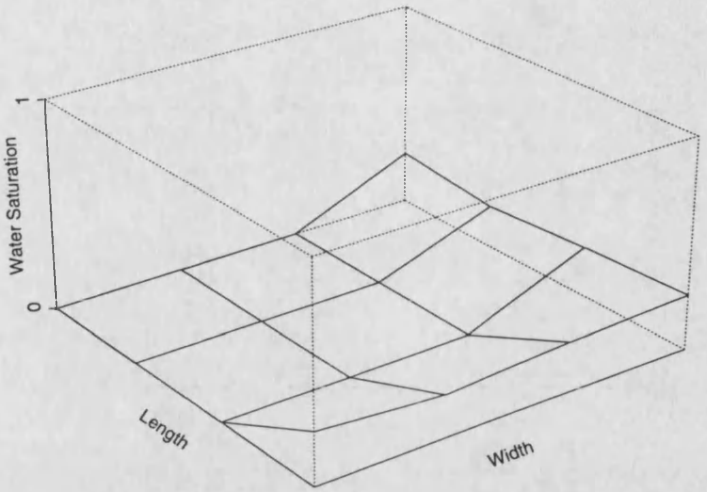


Figure 4-88: Water saturation surface in 2-D reservoir with 4×4 grid-blocks at 150 days.

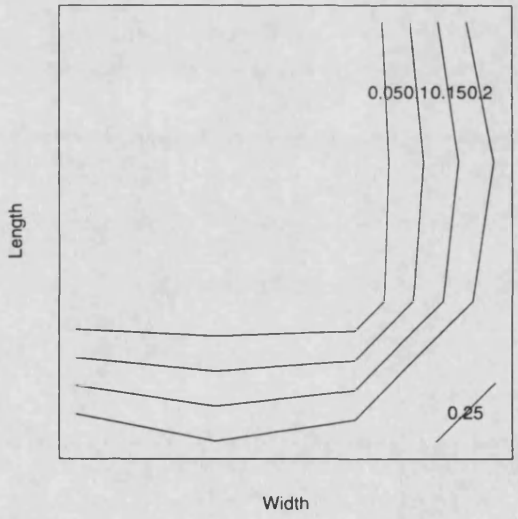


Figure 4-89: Water saturation contour in 2-D reservoir with 4×4 grid-blocks at 150 days.

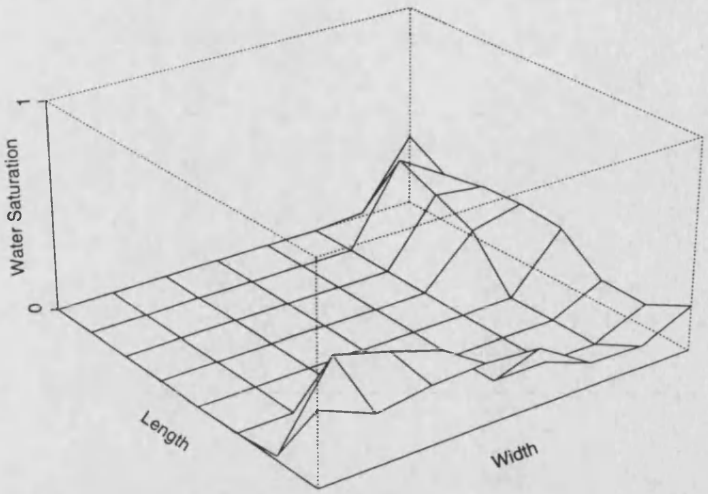


Figure 4-90: Water saturation surface in 2-D reservoir with 8×8 grid-blocks at 150 days.

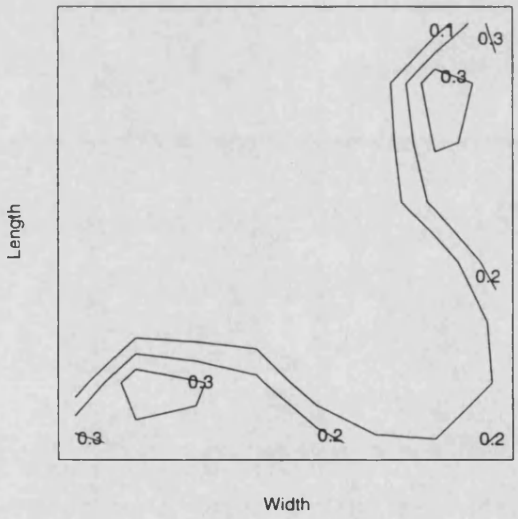


Figure 4-91: Water saturation contour in 2-D reservoir with 8×8 grid-blocks at 150 days.

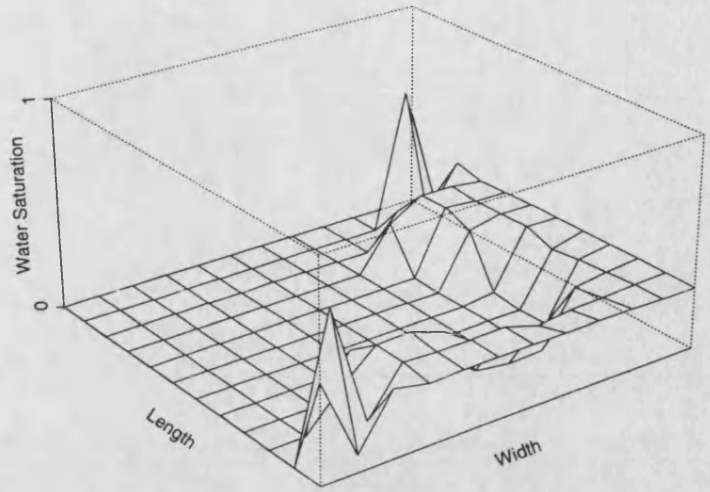


Figure 4-92: Water saturation surface in 2-D reservoir with 12×12 grid-blocks at 150 days.

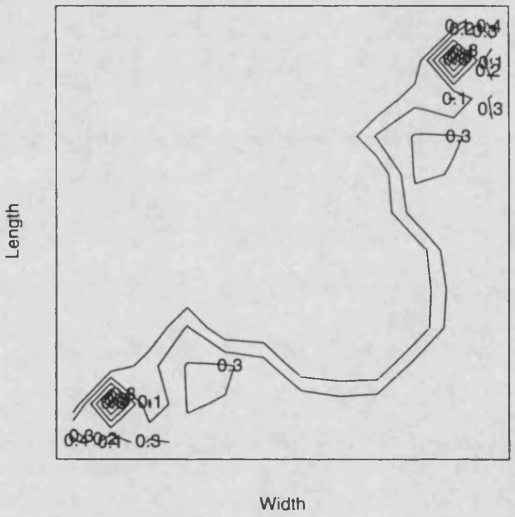


Figure 4-93: Water saturation contour in 2-D reservoir with 12×12 grid-blocks at 150 days.

Chapter 5

The Solution of a System of Linear Equations

In this Chapter we illustrate the validity and efficiency of iterative methods for solving large linear systems arising from the finite-difference discretizations of the equation governing the in-situ combustion process. In particular, we consider the Generalised Minimal Residual Method (GMRES(m)) [46], the Orthogonal Minimisation Method (ORTHOMIN(m)) [59] and the Biconjugate Gradient Stabilised Method (BI-CGSTAB) [56]. Scaling and incomplete decomposition preconditioning of the linear equations are used to improve the convergence properties of the iterative methods.

5.1 Introduction

Systems of linear equations arise in many fields of applied mathematics, for example network equations in electric network theory, numerical solution for ordinary differential equations, partial differential equations and integral equations and numerous other problems. Because of the widespread importance of linear systems, much research has been devoted to their numerical solution.

Let us consider the set of equations,

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{array} \tag{5.1}$$

This system, (5.1), may be written in matrix form as

$$A\mathbf{x} = \mathbf{b}, \tag{5.2}$$

where

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix},$$

and A will be assumed nonsingular, i.e. $detA \neq 0$.

Cramer's rule gives the solution in determinantal form, but this is, of course, not a suitable way to solve numerically, a system of equations with a large number of unknowns, due to the large amount of arithmetic involved in evaluating determinants. In practice the number of operations required for sets of equations of large order is so great that even in high-speed computers the amount of time consumed is prohibitive.

However, there are more suitable numerical methods for solving the linear system (5.2). These methods may be divided into two types, *direct* and *iterative.*

5.1.1 Direct Methods

In a direct method, as the name implies, the algorithm guaranteed to lead to the exact solution after a finite number of mathematical operations in the absence of round-off errors.

The fundamental method used for direct solutions is Gaussian elimination, but even within this class there are various choices of method. For example, LU decomposition and Gauss-Jordan are other well known direct methods [26].

Gaussian elimination is the formal name given to the method of solving systems of linear equations by successively eliminating unknowns and reducing the system to a simpler (triangular) one. In fact to solve (5.2), we reduce it to an equivalent system $\tilde{A}\mathbf{x} = \tilde{\mathbf{b}}$, in which $\tilde{A}$ is upper triangular. This system can be easily solved by a process of back-substitution.

In the Gauss-Jordan method, the elements above the diagonal are made zero as well as those below the diagonal. Usually the diagonal elements are made ones so that this transforms the coefficient matrix, A, into the identity matrix. When this has been accomplished, the column of right-hand sides has been transformed into the solution vector.

As a modification of Gaussian elimination, we have the LU decomposition method. In this method the matrix of coefficients A is transformed into the product of two matrices L and U, i.e., $A = LU$, where L is a lower triangular and U is an upper triangular matrix with ones on its main diagonal. Once the coefficient matrix has been converted to its LU equivalent, we can solve (5.2) as follows. Solve,

$$L\mathbf{z} = \mathbf{b} \tag{5.3}$$

and then find the solution, $\mathbf{x}$, by solving

$$U\mathbf{x} = \mathbf{z}. \tag{5.4}$$

Since L and U are triangular, their solution is straightforward. Solving equation (5.3) requires forward-substitution and (5.4) back-substitution.

Both Gaussian elimination and LU decomposition methods require the same number of arithmetic operations. But, the Gauss-Jordan method requires almost 50 % more operations. The LU decomposition has an important advantage. Equation (5.2) can be solved for any $\mathbf{b}$ without repeating the decomposition.

5.1.2 Iterative Methods

In general, iterative methods may be described as methods which proceed from some initial guess, $\mathbf{x}_0$, and define a sequence of successive approximations $\mathbf{x}_1, \mathbf{x}_2, \ldots$ which, in principle, convergence to the exact solution. If the convergence is sufficiently rapid, the procedure may be terminated at an early stage in the sequence so as to yield a good approximation.

Two important groups of iterative method are, *stationary methods* and *Krylov subspace methods.*

Stationary Methods

If the successive steps are identical, the method is called stationary, otherwise non-stationary. Let us again consider the solution of linear system (5.2). A splitting of the coefficient matrix A is a representation of A in the form

$$A = M - N. \tag{5.5}$$

The problem (5.2) is then equivalent to

$$M\mathbf{x} = N\mathbf{x} + \mathbf{b}.$$

Hence, for nonsingular M, we have the stationary method for constructing a

sequence of approximate solutions to (5.2),

$$\mathbf{x}_{k+1} = M^{-1}N\mathbf{x}_k + M^{-1}\mathbf{b}, \tag{5.6}$$

where $\mathbf{x}_0$ is a initial guess for the solution. The Jacobi, Gauss-Seidel and SOR are examples of such methods [30], [57], [63].

Krylov Subspace Methods

Another group of iterative methods is based on Krylov subspaces.

Definition 5.1 : Given a square matrix A and vector $\mathbf{u}$, let

$$\mathcal{K}_k(\mathbf{u}, A) \equiv span\{\mathbf{u}, A\mathbf{u}, \ldots, A^{k-1}\mathbf{u}\},$$

then $\mathcal{K}_k(\mathbf{u}, A)$ is called the *Krylov subspace* generated by A with respect to $\mathbf{u}$.

Given an initial guess $\mathbf{x}_0$ for (5.2) the vector $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ is called the residual vector; it is, of course, $\mathbf{0}$ for the exact solution.

A Krylov subspace method produces a sequence of iterates of the form

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{v}_k, \tag{5.7}$$

where $\mathbf{v}_k \epsilon \mathcal{K}_k(\mathbf{r}_0, A)$. Any iterate of the form (5.7) satisfies

$$\mathbf{x}_k = \mathbf{x}_0 + \varphi_{k-1}(A)\mathbf{r}_0,$$

where $\varphi_{k-1}(z)$ is a polynomial of degree $k-1$. Equivalently, the residual satisfies

$$\mathbf{r}_k = \phi_k(A)\mathbf{r}_0,$$

where $\phi_k(z)$ is a member of the set $\mathcal{P}_k$, where,

$$\mathcal{P}_k \equiv \{ \text{ polynomials } \phi_k \text{ of degree } k \text{ satisfying } \phi_k(0) = 1 \}.$$

The Conjugate Gradient (CG) and the Conjugate Residual (CR) are basic Krylov subspace methods [31], [51].

5.1.3 Comparison of Direct and Iterative Methods

Matrices associated with linear systems are also classified as dense or sparse.

Unless the coefficients are mostly zero then the linear system is called dense. For such systems, the coefficient matrix A must generally be stored in the main memory of the computer in order to efficiently solve the linear system, and thus memory storage limitations in most computers may limit the order of the system.

A sparse matrix is one in which most coefficients are zero. These systems arise commonly in the numerical solution of partial differential equations.

Because of the large order of most sparse systems of linear equations (sometimes as large as 10^5 or more) the linear system cannot usually be solved efficiently by a direct method such as Gaussian elimination. Iterative methods are the preferred method of solution. They are frequently used to solve problems involving three spatial variables, problems involving nonlinear systems of equations, problems resulting from the discretizations of coupled partial differential equations, and time-dependent problems involving more than one spatial variable.

The important advantages of iterative methods are the simplicity and uniformity of the operations to be performed, which make them well suited for use on computers, also they are relatively insensitive to round-off errors.

Over the past decade, several efficient iterative methods have been developed to solve large sparse systems of linear algebraic equations. The Krylov subspace methods, of which the conjugate gradient method is a well-known example, have proven to be particularly effective for solving the linear systems that arise in the numerical solution of elliptic and parabolic partial differential equations [9].

The Conjugate Gradient (CG) method applied to the symmetric positive def-

inite problem minimises the error functional

$$E_{CG}(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_k\|_A \tag{5.8}$$

along the direction $\mathbf{p}_k$ in order to determine the step length α_k in

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k.$$

Where

$$\alpha_k = (\mathbf{r}_k, \mathbf{r}_k)/(\mathbf{p}_k, A\mathbf{p}_k),$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k,$$

$$\beta_k = (\mathbf{r}_{k+1}, \mathbf{r}_{k+1})/(\mathbf{r}_k, \mathbf{r}_k),$$

and $\mathbf{p}_k$ satisfies

$$(\mathbf{p}_k, A\mathbf{p}_j) = \mathbf{0} \quad for \quad k \neq j.$$

In the expression (5.8) $\| \ \|_A$ is called "A-norm" and is defined as follows.

Definition 5.2 : Suppose A is a symmetric positive definite matrix. The "A-norm" of a vector $\mathbf{u}$ is defined by

$$\|\mathbf{u}\|_A \equiv [(\mathbf{u}, A\mathbf{u})]^{1/2}.$$

The Conjugate Residual (CR) method applied to the symmetric positive definite problem minimises the error functional

$$E_{CR}(\mathbf{x}) = \|\mathbf{r}_{k+1}\|_2$$

along the direction $\mathbf{p}_k$ in order to determine the step length α_k in

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k.$$

Where

$$\alpha_k = (\mathbf{r}_k, A\mathbf{r}_k)/(A\mathbf{p}_k, A\mathbf{p}_k), \tag{5.9}$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k. \tag{5.10}$$

and

$$\beta_k = (\mathbf{r}_{k+1}, A\mathbf{r}_{k+1})/(\mathbf{r}_k, A\mathbf{r}_k). \tag{5.11}$$

and The direction vector $\mathbf{p}_k$ is made A^TA - orthogonal to $\mathbf{p}_{k-1}$, that is

$$(A\mathbf{p}_k, A\mathbf{p}_j) = \mathbf{0} \quad for \quad k \neq j. \tag{5.12}$$

When the system is not symmetric and positive definite then $\| \ \|_A$ does not define a norm so that it does not make sense to minimise error functional $E_{CG}(\mathbf{x})$. Hence the CG method is not applicable.

Similarly if A is not symmetric and positive definite then the expressions (5.9) - (5.12) do not hold and the CR algorithm will break down as well.

A standard technique for solving nonsymmetric systems is to consider the equivalent system

$$A^TA\mathbf{x} = A^T\mathbf{b}$$

and apply techniques available for solving symmetric positive definite systems. However, the condition of the matrix A^TA may be much worse than the condition of A [30].

Recently, many Krylov subspace iterative methods have been derived for solving nonsymmetric linear systems of equations. These methods are generalisation of the CG and CR methods for symmetric and positive definite linear systems.

For example for the CR method, it is not possible to generate an orthogonal basis for $\mathcal{K}_k(\mathbf{r}_0, A)$ using (5.9) and (5.11). However, a basis can be generated by a way which is a modification of Gram-Schmidt procedure. Because of this the search direction $\mathbf{p}_k$ is a combination of all the previously constructed vectors

$\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_{k-1}$.

For example, $\mathbf{p}_k$ can be constructed

$$\mathbf{p}_k = \mathbf{r}_k + \sum_{j=0}^{k} \beta_j^k \mathbf{p}_j,$$

where

$$\beta_j^k = -(A\mathbf{r}_{k+1}, A\mathbf{p}_j)/(A\mathbf{p}_j, A\mathbf{p}_j), \quad j \leq k.$$

This method is called the Generalised Conjugate Residual (GCR) method [19], [20], [22].

But the work and storage requirements of such a computation will become prohibitive for large k. To avoid this difficulty, there are two distinct ways, either by truncating the space or by restarting the algorithm every m step. The Orthogonal Minimisation Method (ORTHOMIN(m)) [59] and the Generalised Minimal Residual Method (GMRES(m)) [46] are two examples for truncated and restarted algorithm respectively.

Other examples for the Krylov subspace iterative methods to solve nonsymmetric linear systems of equations are The Generalised Conjugate Gradient Method (GCG) [14],[61], the Orthogonal Residual Method (ORTHORES) [64], the Biconjugate Gradient Method (BCG) [24],[36], the Conjugate Gradient Squared Method (CGS) [50] and the Biconjugate Gradient Stabilised Method (BI-CGSTAB) [56].

5.2 Comparison of Direct and Iterative Methods on The Presented Simulator

A time consuming part of the calculation in reservoir simulation is the solution of linear systems of the form (5.2) which arise during the use of Newton's method for solving the nonlinear system of the finite-difference equations.

In general, the coefficient matrices in reservoir simulation problems are block

matrices and they are also sparse. For small size problems direct solution may well be an efficient method. For example, for 1D models less than 15 grid-blocks in length, direct solution will be the preferred method. But the simulation of in-situ combustion process in 2D or 3D requires the use of large numbers of grid-blocks for accurate models. As a result of this sets of large, sparse linear systems must be solved. It may well be more economical to solve such a system with iterative rather than direct methods.

However, these matrices are neither symmetric nor necessarily diagonally dominant. Particularly, the in-situ combustion problem represents one of the most difficult types of problem that is encountered in simulation. It produces a highly nonsymmetric matrix because of multiple moving fronts that can change their relative positions during the process [4].

In recent years, experience has shown that iterative methods such as ORTHOMIN [59] and ORTHORES [64] with incomplete decomposition seem to be very robust for the types of problems met in reservoir simulation.

For comparison we consider LU decomposition as a direct method and three other Krylov subspace methods as iterative methods. The iterative methods of interest here are the ORTHOMIN(m) which was developed by Vinsome [59] : it is a generalised conjugate residual method. GMRES(m) is the restarted version of the generalised minimal residual method of Saad and Schultz [46]. BI-CGSTAB is one of the latest iterative methods developed by H.A. Van der Vorst [56].

To increase the convergence rate for the iterative methods a preconditioning technique, Incomplete LU decomposition (ILU), is used.

5.2.1 Methods and Algorithms

We wrote a FORTRAN77 code which implements the methods described in this Chapter. In order to test the accuracy of computer programs for solving numerical problems, one needs numerical examples with known solutions. From

this point each algorithm has been tested with a collection of known solutions of systems of linear equations. Test matrices were selected from the literatures each of them representing different group of matrices such as diagonally dominant, symmetric and nonsymmetric [5], [25], [29].

In the remainder of this section, we give a brief description and an algorithm for the methods to be used in the comparison.

LU Decomposition Method

Let us consider the solution of the system of linear equations of the form (5.2). When the coefficient matrix has been factorized into triangular matrices, L and U, the matrix equation then becomes

$$LU\mathbf{x} = \mathbf{b}$$

which can be solved in two steps. First we solve

$$L\mathbf{z} = \mathbf{b}$$

for a dummy variable $\mathbf{z}$, and then solve

$$U\mathbf{x} = \mathbf{z}$$

for $\mathbf{x}$.

For example, suppose we wish to factor

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

into the factors of the forms

$$L = \begin{pmatrix} 1 & 0 \\ l_{21} & 1 \end{pmatrix} \qquad and \qquad U = \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix}.$$

Their product is

$$LU = \begin{pmatrix} u_{11} & u_{12} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} \end{pmatrix}.$$

If L and U are factors of A, then

$$LU = A$$

and every element in the product LU will be identical to the corresponding element in A. That is,

$$u_{11} = a_{11} \qquad and \qquad l_{21}u_{11} = a_{21}$$

thus

$$l_{21} = a_{21}/a_{11} \qquad and \qquad u_{12} = a_{12}$$

and

$$l_{21}u_{12} + u_{22} = a_{22}$$

thus

$$u_{22} = a_{22} - (a_{21}a_{12}/a_{11}).$$

Hence, the two factors are

$$L = \begin{pmatrix} 1 & 0 \\ a_{21}/a_{11} & 1 \end{pmatrix} \qquad and \qquad U = \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} - (a_{21}a_{12}/a_{11}) \end{pmatrix}.$$

LU decomposition in this form can easily be adapted to pivoting strategy should such strategy be required.

Algorithm 5.1 describes complete procedure to solve (5.2) by *LU* decomposition method.

Algorithm 5.1: *LU* Decomposition with Forward and Backward Substitution.

1. *Compute LU Decomposition.*

   For $i = 1, 2, 3, \ldots, n$ *do:*

   $l_{i,i} = 1$

   Enddo.

   For $j = 1, 2, 3, \ldots, n$ *do:*

   For $i = 1, 2, 3, \ldots, j$ *do:*

   $u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}$

   Enddo.

   For $i = j+1, j+2, \ldots, n$ *do:*

   $l_{ij} = \left[a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}\right] / u_{ii}$

   Enddo.

   Enddo.

2. *Forward Substitution for Solving* $L\mathbf{z} = \mathbf{b}$.

   $z_1 = b_1$

   For $i = 2, 3, \ldots, n$ *do:*

   $z_i = b_i - \sum_{j=1}^{i-1} l_{ij} z_j$

   Enddo.

3. *Backward Substitution for Solving* $U\mathbf{x} = \mathbf{z}$.

   $x_n = z_n / u_{nn}$

   For $i = n-1, n-2, \ldots, 1$ *do:*

   $x_i = \left[z_i - \sum_{j=i+1}^{n} u_{ij} x_j\right] / u_{ii}$

   Enddo.

The Generalised Minimal Residual Method (GMRES(m))

The Generalised Minimal Residual Method (GMRES) is an iterative method for solving large linear systems of equations (5.2) in which A is nonsymmetric.

The GMRES method begins with an initial approximation $\mathbf{x}_0$ and initial residual $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$. At the mth iteration, a correction $\mathbf{z}_m$ is determined in the Krylov subspace $\mathcal{K}_m(\mathbf{r}_0, A)$ which solves the least squares problem

$$min_{\mathbf{z}\epsilon\mathcal{K}_m(\mathbf{r}_0,A)}\|\mathbf{b} - A(\mathbf{x}_0 + \mathbf{z})\|_2.$$

The mth iterate is then $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{z}_m$.

The implementation of GMRES given by Saad and Schultz [46] determines $\mathbf{z}_m$ by maintaining for each m a particular orthonormal basis $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ of the Krylov subspace $\mathcal{K}_m(\mathbf{r}_0, A)$ together with an upper Hessenberg matrix H_m. These are generated through Arnoldi's method, the basic form of which is the following:

Algorithm 5.2: Arnoldi.

1. *Choose an initial vector* $\mathbf{v}_1$ *with* $\|\mathbf{v}_1\|_2 = 1$.
2. *For* $k = 1, 2, 3, \ldots, MAXIT$ *do:*

   For $i = 1, 2, 3, \ldots, k$ *do:*

   $h_{i,k} = (A\mathbf{v}_k, \mathbf{v}_i)$

   Enddo.

   $\mathbf{q}_{k+1} = A\mathbf{v}_k - \sum_{i=1}^{k} h_{i,k}\mathbf{v}_i$

   $h_{k+1,k} = \|\mathbf{q}_{k+1}\|_2$

   $\mathbf{v}_{k+1} = \mathbf{q}_{k+1}/h_{k+1,k}$

   Enddo.

$\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ generated by Arnoldi's method is an l_2-orthonormal basis of $\mathcal{K}_m(\mathbf{r}_0, A)$ for each m.

In the GMRES method, $\mathbf{v}_1 = \mathbf{r}_0/\|\mathbf{r}_0\|_2$ and the h_{ij}'s constitute the nonzero elements of the matrices $\hat{H}_m$.

Let

$$V_m = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m],$$

i.e., V_m is an $n \times m$ matrix, then

$$AV_m = V_{m+1}\hat{H}_m,$$

for each m. Where $\hat{H}_m$ is the same as H_m except for an additional row whose only nonzero element is $h_{m+1,m}$ in the $(m+1, m)$ position.

With these orthonormal bases and upper Hessenberg matrices, $\mathbf{z}_m$ is determined as follows: Setting

$$\mathbf{z} = V_m\mathbf{y}$$

for $\mathbf{y}\epsilon\mathbf{R}^m$ gives

$$min_{\mathbf{z}\epsilon\mathcal{K}_m(\mathbf{r}_0,A)}\|\mathbf{b} - A(\mathbf{x}_0 + \mathbf{z})\|_2 = min_{\mathbf{y}\epsilon\mathbf{R}^m}\|\mathbf{r}_0 - AV_m\mathbf{y}\|_2$$

$$= min_{\mathbf{y}\epsilon\mathbf{R}^m}\|\mathbf{r}_0 - V_{m+1}\hat{H}_m\mathbf{y}\|_2 \qquad (5.13)$$

Since $\mathbf{v}_1 = \mathbf{r}_0/\|\mathbf{r}_0\|_2$ and since $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{m+1}\}$ is orthonormal, one has

$$min_{\mathbf{y}\epsilon\mathbf{R}^m}\|\mathbf{r}_0 - V_{m+1}\hat{H}_m\mathbf{y}\|_2 = min_{\mathbf{y}\epsilon\mathbf{R}^m}\|\beta\mathbf{e}_1 - \hat{H}_m\mathbf{y}\|_2 \qquad (5.14)$$

where $\beta = \|\mathbf{r}_0\|_2$ and $\mathbf{e}_1$ is the unit vector $\mathbf{e}_1 = (1, 0, \ldots, 0)^T$ in $\mathbf{R}^{m+1}$.

A QR decomposition of $\hat{H}_m$ is performed, i.e. $Q_m\hat{H}_m = R_m$, and the upper triangular matrix R_m is used solve a system of the type $R_m\mathbf{y} = \mathbf{g}$, to yield the minimisation solution $\mathbf{y}_m$. The solution to the linear system is finally computed as

$$\mathbf{x}_m = \mathbf{x}_0 + V_m\mathbf{y}_m.$$

Briefly, in the GMRES method, an orthonormal basis of directions is formed and the solution is given by a linear combination of the basis vectors so as to minimise the norm of the residual in the subspace spanned by the basis. Because storing all the previous directions is very costly, in practice a truncated version GMRES(m) is used. That is after m steps we restart the GMRES process using the current approximation as the initial vector.

Algorithm 5.3 : GMRES(m)

1. *Choose* $\mathbf{x}_0$.
2. *Compute* $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$.
3. *Compute* $\beta = \|\mathbf{r}_0\|_2$.
4. *Compute* $\mathbf{v}_1 = \mathbf{r}_0/\beta$.
5. *For* $k = 0, 1, 2, \ldots, MAXIT$ *do:*

   For $j = 1, 2, 3, \ldots, m$ *do:*

   For $i = 1, 2, 3, \ldots, j$ *do:*

   $h_{i,j} = (A\mathbf{v}_j, \mathbf{v}_i)$

   Enddo.

   $\mathbf{q}_{j+1} = A\mathbf{v}_j - \sum_{i=1}^{j} h_{i,j}\mathbf{v}_i$

   $h_{j+1,j} = \|\mathbf{q}_{j+1}\|_2$

   $\mathbf{v}_{j+1} = \mathbf{q}_{j+1}/h_{j+1,j}$

   Enddo.

6. *Form the approximate solution:*

   $\mathbf{x}_{k+1} = \mathbf{x}_k + V_m\mathbf{y}_m$, *where* $\mathbf{y}_m$ *minimises* $\|\beta\mathbf{e}_1 - \hat{H}_m\mathbf{y}\|_2$, $\mathbf{y} \epsilon R^m$.

7. *Compute* $\mathbf{r}_{k+1} = \mathbf{b} - A\mathbf{x}_{k+1}$.
8. *Compute* $\|\mathbf{r}_{k+1}\|_2$ *: if satisfied then stop* .
9. *Else: set* $\beta = \|\mathbf{r}_{k+1}\|_2$.
10. *Compute* $\mathbf{v}_1 = \mathbf{r}_{k+1}/\beta$.

Enddo.

The Orthogonal Minimisation Method (ORTHOMIN(m))

For symmetric positive definite problems, the conjugate residual method computes a sequence of approximate solutions by an iteration of the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k,$$

and the step-length α_k minimises

$$\|\mathbf{b} - A(\mathbf{x}_k + \alpha_k \mathbf{p}_k)\|_2 = \|\mathbf{r}_{k+1}\|_2,$$

as a function of α_k. The direction vectors $\mathbf{p}_k$ are computed by a two term recurrence of the form

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k,$$

and they satisfy

$$(A\mathbf{p}_k, A\mathbf{p}_j) = 0, \quad k \neq j. \tag{5.15}$$

In the nonsymmetric case, a set of directions that satisfy (5.15) can be computed as follows,

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \sum_{j=max(0,k-m+1)}^{k} \beta_j^k \mathbf{p}_j,$$

where

$$\beta_j^k = -(A\mathbf{r}_{k+1}, A\mathbf{p}_j)/(A\mathbf{p}_j, A\mathbf{p}_j), \quad j \leq k. \tag{5.16}$$

This method is called as ORTHOMIN(m).

Algorithm 5.4 : ORTHOMIN(m)

1. *Choose* $\mathbf{x}_0 = \mathbf{0}$.
2. *Compute* $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$.
3. *Set* $\mathbf{p}_0 = \mathbf{r}_0$.
4. *For* $k = 0, 1, 2, \ldots, MAXIT$ *do:*

$\alpha_k = (\mathbf{r}_k, A\mathbf{p}_k)/(A\mathbf{p}_k, A\mathbf{p}_k)$

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$

$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{p}_k$

Compute $\| \mathbf{r}_{k+1} \|_2$ *; if satisfied then stop .*

$j_k = max(0, k - m + 1)$

For $j = j_k, \ldots, k$ *do:*

$\beta_j^k = -(A\mathbf{r}_{k+1}, A\mathbf{p}_j)/(A\mathbf{p}_j, A\mathbf{p}_j)$

Enddo.

$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \sum_{j=j_k}^{k} \beta_j^k \mathbf{p}_j$

$A\mathbf{p}_{k+1} = A\mathbf{r}_{k+1} + \sum_{j=j_k}^{k} \beta_j^k A\mathbf{p}_j$

Enddo.

The Biconjugate Gradient Stabilised Method (BI-CGSTAB)

The Biconjugate Gradient Stabilised Method (BI-CGSTAB) is another Krylov subspace method, which has recently been developed by Van der Vorst [56] for solving nonsymmetric linear systems. It is closely related to the Biconjugate Gradient Method (BCG) and the Conjugate Gradient Squared Method (CGS). The main idea is to form a product of the BCG polynomial with another, locally defined polynomial. It is a more smoothly converging variant of CGS.

Here, BCG and CGS algorithms are defined to establish notation and terminology.

Algorithm 5.5 : BCG

1. *Choose* $\mathbf{x}_0$.
2. *Compute* $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$.
3. *Set* $\tilde{\mathbf{r}}_0 = \mathbf{r}_0$

   $\tilde{\mathbf{p}}_0 = \mathbf{p}_0 = \mathbf{0}$

   $\rho_0 = 1$
4. *For* $k = 1, 2, 3, \ldots, MAXIT$ *do:*

$\rho_k = (\tilde{\mathbf{r}}_{k-1}, \mathbf{r}_{k-1})$

$\beta_k = (\rho_k / \rho_{k-1})$

$\mathbf{p}_k = \mathbf{r}_{k-1} + \beta_k \mathbf{p}_{k-1}$

$\tilde{\mathbf{p}}_k = \tilde{\mathbf{r}}_{k-1} + \beta_k \tilde{\mathbf{p}}_{k-1}$

$\mathbf{v}_k = A\mathbf{p}_k$

$\alpha_k = \rho_k / (\tilde{\mathbf{p}}_k, \mathbf{v}_k)$

$\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k \mathbf{p}_k$

$\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_k \mathbf{v}_k$

Compute $\|\mathbf{r}_k\|_2$ *: if satisfied then stop .*

$\tilde{\mathbf{r}}_k = \tilde{\mathbf{r}}_{k-1} - \alpha_k A^T \tilde{\mathbf{p}}_k$

Enddo.

Algorithm 5.6 : CGS

1. *Choose* $\mathbf{x}_0$.
2. *Compute* $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$.
3. *Set* $\tilde{\mathbf{r}}_0 = \mathbf{r}_0$

   $\mathbf{p}_0 = \mathbf{q}_0 = \mathbf{0}$

   $\rho_0 = 1$
4. *For* $k = 1, 2, 3, \ldots, MAXIT$ *do:*

   $\rho_k = (\tilde{\mathbf{r}}_0, \mathbf{r}_{k-1})$

   $\beta = \rho_k / \rho_{k-1}$

   $\mathbf{u} = \mathbf{r}_{k-1} + \beta \mathbf{q}_{k-1}$

   $\mathbf{p}_k = \mathbf{u} + \beta(\mathbf{q}_{k-1} + \beta \mathbf{p}_{k-1})$

   $\mathbf{v} = A\mathbf{p}_k$

   $\alpha = \rho_k / (\tilde{\mathbf{r}}_0, \mathbf{v})$

   $\mathbf{q}_k = \mathbf{u} - \alpha \mathbf{v}$

   $\mathbf{w} = \mathbf{u} + \mathbf{q}_k$

   $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha \mathbf{w}$

$\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha A\mathbf{w}$

Compute $\|\mathbf{r}_k\|_2$ *: if satisfied then stop .*

Enddo.

From the definition of the Krylov subspace methods and Algorithms 5.5 and 5.6, the residual vectors $\mathbf{r}_k$, generated by the BCG and CGS methods, satisfy the relations

$$\mathbf{r}_k = \phi_k(A)\mathbf{r}_0$$

and

$$\mathbf{r}_k = \phi_k^2(A)\mathbf{r}_0.$$

respectively. Where $\phi_k(A)$ is kth degree polynomials in A.

We can construct another iterative method so that

$$\mathbf{r}_k = \psi_k(A)\phi_k(A)\mathbf{r}_0,$$

where ψ_k is taken to be of the form

$$\psi_k(\mathbf{z}) = (1 - \omega_1\mathbf{z})(1 - \omega_2\mathbf{z})\cdots(1 - \omega_k\mathbf{z}), \tag{5.17}$$

for suitably chosen constants ω_j.

From Algorithm 5.5 it can be shown that

$$\mathbf{p}_{k+1} = \varphi_k(A)\mathbf{r}_0$$

in which $\varphi_k(A)$ is a kth degree polynomials in A. The Algorithm 5.5 also defines the relations between the polynomials $\varphi_k(A)$ and $\phi_k(A)$:

$$\varphi_k(A)\mathbf{r}_0 = [\phi_k(A) + \beta_{k+1}\varphi_{k-1}(A)]\,\mathbf{r}_0$$

and

$$\phi_k(A)\mathbf{r}_0 = [\phi_{k-1}(A) - \alpha_{k+1}A\varphi_{k-1}(A)]\,\mathbf{r}_0.$$

In the BI-CGSTAB Algorithm we need to have recurrence relations for

$$\mathbf{r}_k = \psi_k(A)\phi_k(A)\mathbf{r}_0.$$

Let us take ψ_k as in (5.17). Because of the BCG relation for the factor ϕ_k and φ_k,

$$\psi_k(A)\phi_k(A)\mathbf{r}_0 = (1 - \omega_k A)\psi_{k-1}(A)\,[\phi_{k-1}(A) - \alpha_k A\varphi_{k-1}(A)]\,\mathbf{r}_0$$

$$= [\psi_{k-1}(A)\phi_{k-1}(A) - \alpha_k A\psi_{k-1}\varphi_{k-1}(A)]\,\mathbf{r}_0 - \omega_k A\,[\psi_{k-1}(A)\phi_{k-1}(A) - \alpha_k A\psi_{k-1}\varphi_{k-1}(A)]\,\mathbf{r}_0.$$

We also need a relation for the product $\psi_k(A)\varphi_k(A)\mathbf{r}_0$. This can also be obtained from the BCG relations:

$$\psi_k(A)\varphi_k(A)\mathbf{r}_0 = \psi_k(A)\,[\phi_k(A) + \beta_{k+1}\varphi_{k-1}(A)]\,\mathbf{r}_0$$

$$= \psi_k(A)\phi_k(A)\mathbf{r}_0 + \beta_{k+1}(1 - \omega_k A)\psi_{k-1}(A)\varphi_{k-1}(A)\mathbf{r}_0$$

$$= \psi_k(A)\phi_k(A)\mathbf{r}_0 + \beta_{k+1}\psi_{k-1}(A)\varphi_{k-1}(A)\mathbf{r}_0 - \beta_{k+1}\omega_k A\psi_{k-1}(A)\varphi_{k-1}(A)\mathbf{r}_0.$$

By replacing $\mathbf{r}_k$ with $\psi_k(A)\phi_k(A)\mathbf{r}_0$ and $\mathbf{p}_k$ with $\psi_{k-1}(A)\varphi_{k-1}(A)\mathbf{r}_0$, we find the following Algorithm.

Algorithm 5.7 : BI-CGSTAB

1. *Choose* $\mathbf{x}_0$.
2. *Compute* $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$.
3. *Set* $\tilde{\mathbf{r}}_0 = \mathbf{r}_0$

   $\mathbf{v}_0 = \mathbf{p}_0 = \mathbf{0}$

   $\rho_0 = \omega_0 = \alpha = 1$
4. *For* $k = 1, 2, 3, \ldots, MAXIT$ *do:*

$\rho_k = (\tilde{\mathbf{r}}_0, \mathbf{r}_{k-1})$

$\beta = (\rho_k/\rho_{k-1})(\alpha/\omega_{k-1})$

$\mathbf{p}_k = \mathbf{r}_{k-1} + \beta(\mathbf{p}_{k-1} - \omega_{k-1}\mathbf{v}_{k-1})$

$\mathbf{v}_k = A\mathbf{p}_k$

$\sigma_k = (\tilde{\mathbf{r}}_0, \mathbf{v}_k)$

$\alpha = \rho_k/\sigma_k$

$\mathbf{s} = \mathbf{r}_{k-1} - \alpha\mathbf{v}_k$

$\mathbf{t} = A\mathbf{s}$

$\omega_k = (\mathbf{t}, \mathbf{s})/(\mathbf{t}, \mathbf{t})$

$\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha\mathbf{p}_k + \omega_k\mathbf{s}$

$\mathbf{r}_k = \mathbf{s} - \omega_k\mathbf{t}$

Compute $\|\mathbf{r}_k\|_2$ *: if satisfied then stop .*

Enddo.

5.2.2 Preconditioning and Scaling

The rate of convergence of iterative methods can usually be speeded up by the technique of preconditioning.

Let M be a nonsingular matrix, then the basic principle of preconditioning is that we replace the original linear system (5.2) by the equivalent system

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b},$$

where M is a matrix such that $M^{-1}\mathbf{w}$ is inexpensive to compute for any vector $\mathbf{w}$ and $M^{-1}A$ is close to the identity in some sense.

Incomplete LU decomposition (ILU) is one of the most popular preconditioning technique for the iterative methods.

It has been successfully used by many authors in the symmetric positive definite case [34]. It appears also very attractive in the nonsymmetric case, as reported in [22], [35], [37], [55].

Incomplete LU Decomposition (ILU)

During the steps of a direct solution procedure additional nonzero elements are generated, called fill-in terms. Fill-in increases storage requirements and computational work. Because of fill-in, Matrices L and U in the decomposition $A = LU$ are much less sparse than the original matrix A.

The basic idea of Incomplete LU (ILU) decomposition is to preserve the sparsity in the L and U factors by neglecting most of, or even all, the fill-in terms during the elimination procedure. Of course, the product LU is then only an approximation of A i.e.

$$A \approx M = LU,$$

where L is a lower triangular matrix and U is an upper triangular matrix.

ILU preconditioning has been used here for all the iterative schemes and is defined as follows :

Let $\mathcal{N} \subseteq \{(i,j) | 1 \leq i,j \leq n\}$. In case of 1D, $\mathcal{N}$ contains the indices of all nonzero elements in the original matrix which are on main, first upper and first lower block diagonals.

In case of 2D, $\mathcal{N}$ contains the indices of the elements on the five block diagonals.

Algorithm 5.8: Incomplete LU Decomposition

For $i = 1, 2, 3, \ldots, n$ *do:*

  For $j = 1, 2, 3, \ldots, n$ *do:*

    If $(i,j) \in \mathcal{N}$ *then*

      $s_{ij} = a_{ij} - \sum_{k=1}^{min(i,j)-1} l_{ik} u_{kj}$

      If $i \geq j$ *then* $l_{ij} = s_{ij}$

      If $i < j$ *then* $u_{ij} = s_{ij}/l_{i,i}$

Endif.

Enddo.

$u_{ii} = 1$

For $j = i+1, i+2, i+3, \ldots, n$ *do:*

$u_{ij} = u_{ij}/l_{ii}$

Enddo.

Enddo.

Now the preconditioned forms of the Algorithms of the three iterative methods can be written as follows.

Algorithm 5.9 : Preconditioned GMRES(m)

1. *Choose* $\mathbf{x}_0$.
2. *Compute* $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$.
3. *Compute* $\beta = \| \mathbf{r}_0 \|_2$.
4. *Compute* $\mathbf{v}_1 = \mathbf{r}_0/\beta$.
5. *For* $k = 0, 1, 2, \ldots, MAXIT$ *do:*

For $j = 1, 2, 3, \ldots, m$ *do:*

$\mathbf{z} = M^{-1}\mathbf{v}_j$

$\mathbf{w} = A\mathbf{z}$

For $i = 1, 2, 3, ..., j$ *do:*

$h_{i,j} = (\mathbf{w}, \mathbf{v}_i)$

$\mathbf{w} = \mathbf{w} - h_{i,j}\mathbf{v}_i$

Enddo.

$h_{j+1,j} = \|\mathbf{w}\|_2$

$\mathbf{v}_{j+1} = \mathbf{w}/h_{j+1,j}$

Enddo.

6. *Form the approximate solution:*

$\mathbf{x}_{k+1} = \mathbf{x}_k + M^{-1}V_m\mathbf{y}_m$,

where $\mathbf{y}_m$ *minimises* $\| \beta\mathbf{e}_1 - \hat{H}_m\mathbf{y} \|_2$, $\mathbf{y}\epsilon R^m$.

7. *Compute* $\mathbf{r}_{k+1} = \mathbf{b} - A\mathbf{x}_{k+1}$.
8. *Compute* $\|\mathbf{r}_{k+1}\|_2$ *: if satisfied then stop .*
9. *Else: set* $\beta = \| \mathbf{r}_{k+1} \|_2$.
10. *Compute* $\mathbf{v}_1 = \mathbf{r}_{k+1}/\beta$.

Enddo.

Algorithm 5.10 : Preconditioned ORTHOMIN(m)

1. *Choose* $\mathbf{x}_0 = \mathbf{0}$.
2. *Compute* $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$.
3. *Set* $\mathbf{p}_0 = \mathbf{r}_0$.
4. *For* $k = 0, 1, 2, \ldots, MAXIT$ *do:*

$\alpha_k = (\mathbf{r}_k, A\mathbf{p}_k)/(A\mathbf{p}_k, A\mathbf{p}_k)$

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$

$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{p}_k$

Compute $\|\mathbf{r}_{k+1}\|_2$ *; if satisfied then stop .*

$\mathbf{z} = M^{-1}\mathbf{r}_{k+1}$

$j_k = max(0, k - m + 1)$

For $j = j_k, \ldots, k$ *do:*

$\beta_j^k = -(A\mathbf{z}, A\mathbf{p}_j)/(A\mathbf{p}_j, A\mathbf{p}_j)$

Enddo.

$\mathbf{p}_{k+1} = \mathbf{z} + \sum_{j=j_k}^{k} \beta_j^k \mathbf{p}_j$

$A\mathbf{p}_{k+1} = A\mathbf{z} + \sum_{j=j_k}^{k} \beta_j^k A\mathbf{p}_j$

Enddo.

Algorithm 5.11 : Preconditioned BI-CGSTAB

1. *Choose* $\mathbf{x}_0$.
2. *Compute* $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$.
3. *Set* $\tilde{\mathbf{r}}_0 = \mathbf{r}_0$

   $\mathbf{v}_0 = \mathbf{p}_0 = \mathbf{0}$

   $\rho_0 = \omega_0 = \alpha = 1$
4. *For* $k = 1, 2, 3, \ldots, MAXIT$ *do:*

   $\rho_k = (\tilde{\mathbf{r}}_0, \mathbf{r}_{k-1})$

   $\beta = (\rho_k/\rho_{k-1})(\alpha/\omega_{k-1})$

   $\mathbf{p}_k = \mathbf{r}_{k-1} + \beta(\mathbf{p}_{k-1} - \omega_{k-1}\mathbf{v}_{k-1})$

   $\mathbf{y} = M^{-1}\mathbf{p}_k$

   $\mathbf{v}_k = A\mathbf{y}$

   $\sigma_k = (\tilde{\mathbf{r}}_0, \mathbf{v}_k)$

   $\alpha = \rho_k/\sigma_k$

   $\mathbf{s} = \mathbf{r}_{k-1} - \alpha\mathbf{v}_k$

   $\mathbf{z} = M^{-1}\mathbf{s}$

   $\mathbf{t} = A\mathbf{z}$

   $\omega_k = (\mathbf{t}, \mathbf{s})/(\mathbf{t}, \mathbf{t})$

   $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha\mathbf{y} + \omega_k\mathbf{z}$

   $\mathbf{r}_k = \mathbf{s} - \omega_k\mathbf{t}$

   Compute $\|\mathbf{r}_k\|_2$ *: if satisfied then stop .*

   Enddo.

A scaling of the set of equations is also tested in the system. The approach is to define

$$s_i = max_{1 \le j \le n}|a_{ij}|, \qquad i = 1, 2, 3, \ldots n,$$

$$\hat{a}_{i,j} = a_{ij}/s_i, \qquad i, j = 1, 2, 3, \ldots n,$$

$$\hat{b}_i = b_i/s_i, \qquad i = 1, 2, 3, ...n,$$

where $\hat{A} = [\hat{a}_{ij}]$ and $\hat{b} = [\hat{b}_i]$ are the result of scaling.

The convergence criterion imposed on the residual was varied from the one dimensional case to two dimensional depending on the value of the residual calculated when using the direct method. The object was to make comparable results which are obtained by direct and iterative methods.

All the iterative methods use a vector of zeros as an initial guess.

5.2.3 Results

In this section, we describe the results of numerical experiments in which the methods discussed above are used. A number of tests were performed to compare *LU* decomposition, GMRES, ORTHOMIN and BI-CGSTAB. All methods were tested on three two-dimensional problems. We used the same data file as given in Appendix D except we performed the computations on various grid-blocks: 4×4, 8×8 and 12×12. Tables 5-1, 5-2, and 5-3 show the results for *LU* decomposition, BI-CGSTAB, GMRES(m) and ORTHOMIN(m) with various values for m.

The flags used in the tables are defined as follows:

RT : Reservoir Time (sec),

LIPTS : Linear Iterations Per Time Step,

NIPTS : Newton Iterations Per Time Step,

LIPNI : Linear Iterations Per Newton Iteration,

TOTLI : Total Linear Iterations,

TOTNI : Total Newton Iterations,

WORK : Work Required for all time steps.

We measure the work for each iteration in terms of the number of arithmetic operations, multiplication and division, required to perform it. We consider all iterative methods with preconditioning and scaling.

Our stopping criterion for inner iteration was $\|\mathbf{r}\|_2 \leq TOL$ or the maximum number of iterations ($MAXIT$) has been reached. Where TOL and $MAXIT$ can be specified in the data file.

In the first problem a 4×4 grid-block was used, so order of the matrix was 96. All methods reached the same reservoir time in five time steps and taken an average of 16.8 Newton iterations per time step except GMRES(1). It required average of 17.2 Newton iterations to reach the convergence.

METHODS	RT	LIPTS	NIPTS	LIPNI	TOTLI	TOTNI	WORK
LU Decomposition	361	—	16.8	—	—	84	225×10^4
GMRES(1)	361	600.2	17.2	34.90	3001	86	211×10^5
GMRES(2)	361	145.6	16.8	8.67	728	84	556×10^4
GMRES(3)	361	93.6	16.8	5.57	468	84	381×10^4
GMRES(4)	361	78.8	16.8	4.69	394	84	333×10^4
GMRES(5)	361	72.4	16.8	4.31	362	84	314×10^4
GMRES(10)	361	72.6	16.8	4.32	363	84	331×10^4
GMRES(20)	361	72.6	16.8	4.32	363	84	366×10^4
ORTHOMIN(1)	361	362.6	16.8	21.58	1813	84	894×10^4
ORTHOMIN(2)	361	270.0	16.8	16.07	1350	84	724×10^4
ORTHOMIN(3)	361	237.0	16.8	14.11	1185	84	679×10^4
ORTHOMIN(4)	361	218.4	16.8	13.00	1092	84	663×10^4
ORTHOMIN(5)	361	203.4	16.8	12.11	1017	84	652×10^4
ORTHOMIN(10)	361	166.6	16.8	9.92	833	84	668×10^4
ORTHOMIN(20)	361	166.6	16.8	9.92	833	84	908×10^4
BI-CGSTAB	361	96.2	16.8	5.73	481	84	487×10^4

Table 5-1: Results for the two-dimensional (4×4) simulation.

As a result of the size of the problem, *LU* decomposition required less work than the iterative methods. In general, among the iterative methods GMRES, particularly GMRES(5), seems better than other methods. The performance of GMRES(m) and ORTHOMIN(m) depends upon the chosen value of m.

Both methods required the same number of linear iterations for $m = 10$ and $m = 20$ to reach convergence. BI-CGSTAB is better than ORTHOMIN as shown in Table 5-1.

METHODS	RT	LIPTS	NIPTS	LIPNI	TOTLI	TOTNI	WORK
LU Decomposition	260	—	18.6	—	—	93	108×10^6
GMRES(1)	260	719.8	18.4	39.12	3599	92	114×10^6
GMRES(2)	260	295.8	18.6	15.90	1479	93	529×10^5
GMRES(3)	260	182.4	18.6	9.81	912	93	368×10^5
GMRES(4)	260	250.4	18.4	13.61	1252	92	472×10^5
GMRES(5)	260	154.0	18.6	8.28	770	93	331×10^5
GMRES(10)	260	169.4	18.6	9.11	847	93	370×10^5
GMRES(20)	260	169.8	18.6	9.13	849	93	403×10^5
ORTHOMIN(1)	260	625.0	18.6	35.05	3260	93	721×10^5
ORTHOMIN(2)	260	509.4	16.7	30.52	6622	217	160×10^6
ORTHOMIN(3)	260	440.4	18.6	23.68	2202	93	574×10^5
ORTHOMIN(4)	260	429.8	18.6	23.11	2149	93	587×10^5
ORTHOMIN(5)	260	405.0	18.6	21.77	2025	93	583×10^5
ORTHOMIN(10)	260	3659.4	19.2	190.59	18297	96	516×10^6
ORTHOMIN(20)	260	291.0	18.6	15.65	1455	93	701×10^5
BI-CGSTAB	260	177.6	18.6	9.55	888	93	426×10^5

Table 5-2: Results for the two-dimensional (8×8) simulation.

The second example is a 8×8 grid-block case. Table 5-2 illustrates the typical

convergence behaviour of each of the iterative methods. In this case iterative methods required less work than the direct method *LU* decomposition however, ORTHOMIN(2) required a very large number of Newton iterations. BI-CGSTAB is better than ORTHOMIN and looks competitive with GMRES but GMRES(5) is the best of all.

Finally, we can see from Table 5-3 that for a 12×12 grid-block the advantage of iterative methods is very clear when we consider the amount of work which is required.

METHODS	RT	LIPTS	NIPTS	LIPNI	TOTLI	TOTNI	WORK
LU Decomposition	72	—	17.6	—	—	88	114×10^7
GMRES(1)	72	515.4	17.6	29.23	2572	88	222×10^6
GMRES(2)	72	255.0	17.6	14.49	1275	88	137×10^6
GMRES(3)	72	152.0	17.6	8.64	760	88	103×10^6
GMRES(4)	72	132.8	17.6	7.55	664	88	973×10^5
GMRES(5)	72	118.6	17.6	6.74	593	88	930×10^5
GMRES(10)	72	129.2	17.6	7.34	646	88	993×10^5
GMRES(20)	72	129.8	17.6	7.38	649	88	105×10^6
ORTHOMIN(1)	72	588.0	17.6	33.41	2940	88	180×10^6
ORTHOMIN(2)	72	314.1	14.7	21.34	4397	206	350×10^6
ORTHOMIN(3)	72	421.4	17.6	23.94	2107	88	155×10^6
ORTHOMIN(4)	72	419.4	17.6	23.83	2097	88	160×10^6
ORTHOMIN(5)	72	422.6	17.6	24.01	2113	88	166×10^6
ORTHOMIN(10)	72	2032.0	16.1	126.12	18288	145	130×10^7
ORTHOMIN(20)	72	1946.8	18.4	105.80	9734	92	949×10^6
BI-CGSTAB	72	197.8	17.6	11.24	989	88	134×10^6

Table 5-3: Results for the two-dimensional (12×12) simulation.

In this example ORTHOMIN performed far less well than GMRES and BI-CGSTAB, especially when 10 and 20 orthogonal vectors were used. The performances of GMRES for $m = 4, 5, 10$ are very close but, GMRES(5) is cheaper than the others.

From the above results, if the system is large then iterative methods GMRES, ORTHOMIN and BI-CGSTAB provide a valuable tool for solving linear systems arising from simulation of in-situ combustion. Among the iterative methods ORTHOMIN seems to be less robust and required more work than the others. BI-CGSTAB performed better than ORTHOMIN but is not competitive with GMRES and particularly GMRES(5) is certainly the cheapest of the methods tried here.

Chapter 6

Conclusions and Future Work

A two-dimensional simulator was developed to simulate the in-situ combustion oil recovery process.

The model incorporates heat generation by combustion, heat transfer by conduction and convection within the reservoir, and also heat loss by conduction to an adjacent formation. Furthermore the model includes gravity and capillary pressure effects.

Validation was made using the other two simulators. The model results agreed very well with those calculated by ISCOM [28] and Crookston *et al.* [15] for a one-dimensional bench mark simulation. However, work to determine whether or not the model predicts experimental and field histories has not been conducted. This needs to be investigated in the future.

The present model is applicable to four chemical reactions and six components. The model should be further developed to take into account any number of chemical reactions and any number of components.

As of now the model can handle grid-blocks of equal spacings only. Future work should be directed also towards modifying the program to consider variable grid spacings to allow for better tracking of the movement of the combustion front.

A number of runs were also made to study the sensitivity of results to grid-

block size. The results have been tested with regard to oil saturation, water saturation and temperature profiles. It appears that a good definition of those profiles customarily requires fine grid spacing.

Extension of the present model to radial and three dimensions should not present insurmountable difficulties. However, the computing storage requirement and time may increase substantially, and investigative work is needed here.

The fully implicit finite-difference equations are solved for each time-step by means of a Newton-Raphson procedure. The resulting large and sparse sets of linear equations are solved both by direct and iterative methods.

LU decomposition and three iterative methods, GMRES, ORTHOMIN and BI-CGSTAB have been compared with each other in a number of one and two dimensional test problems.

Tests have shown that GMRES method is highly competitive compared with other iterative methods. We believe that this advantage will be even greater in simulations with problems involving three spatial dimensions. Therefore, this work should be extended to this situation.

Appendix A

Discretization of the Partial Differential Equations

A.1 The Discretization of Water Mass Conservation Equation

$$\frac{6.328}{\Delta x}\left[\left(\frac{\rho_g K K_{rg} Y_1}{\mu_g}\right)_{i,j}^{n+1}\left(\frac{P_{g_{i+1,j}}-P_{g_{i,j}}}{\Delta x}\right)^{n+1}-\left(\frac{\rho_g K K_{rg} Y_1}{\mu_g}\right)_{i-1,j}^{n+1}\left(\frac{P_{g_{i,j}}-P_{g_{i-1,j}}}{\Delta x}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta x}\left[\left(\frac{\rho_g^2 K K_{rg} Y_1 M_g}{\mu_g}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}-\left(\frac{\rho_g^2 K K_{rg} Y_1 M_g}{\mu_g}\right)_{i-1,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}\right]$$

$$+\frac{6.328}{\Delta x}\left[\left(\frac{\rho_w K K_{rw}}{\mu_w}\right)_{i,j}^{n+1}\left(\frac{P_{w_{i+1,j}}-P_{w_{i,j}}}{\Delta x}\right)^{n+1}-\left(\frac{\rho_w K K_{rw}}{\mu_w}\right)_{i-1,j}^{n+1}\left(\frac{P_{w_{i,j}}-P_{w_{i-1,j}}}{\Delta x}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta x}\left[\left(\frac{\rho_w^2 K K_{rw} M_w}{\mu_w}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}-\left(\frac{\rho_w^2 K K_{rw} M_w}{\mu_w}\right)_{i-1,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}\right]$$

$$+\frac{6.328}{\Delta y}\left[\left(\frac{\rho_g K K_{rg} Y_1}{\mu_g}\right)^{n+1}_{i,j}\left(\frac{P_{g_{i,j+1}}-P_{g_{i,j}}}{\Delta y}\right)^{n+1}-\left(\frac{\rho_g K K_{rg} Y_1}{\mu_g}\right)^{n+1}_{i,j-1}\left(\frac{P_{g_{i,j}}-P_{g_{i,j-1}}}{\Delta y}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta y}\left[\left(\frac{\rho_g^2 K K_{rg} Y_1 M_g}{\mu_g}\right)^{n+1}_{i,j}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}-\left(\frac{\rho_g^2 K K_{rg} Y_1 M_g}{\mu_g}\right)^{n+1}_{i,j-1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}\right]$$

$$+\frac{6.328}{\Delta y}\left[\left(\frac{\rho_w K K_{rw}}{\mu_w}\right)^{n+1}_{i,j}\left(\frac{P_{w_{i,j+1}}-P_{w_{i,j}}}{\Delta y}\right)^{n+1}-\left(\frac{\rho_w K K_{rw}}{\mu_w}\right)^{n+1}_{i,j-1}\left(\frac{P_{w_{i,j}}-P_{w_{i,j-1}}}{\Delta y}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta y}\left[\left(\frac{\rho_w^2 K K_{rw} M_w}{\mu_w}\right)^{n+1}_{i,j}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}-\left(\frac{\rho_w^2 K K_{rw} M_w}{\mu_w}\right)^{n+1}_{i,j-1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}\right]$$

$$+\left[q_1+(s_3 r_A+s_6 r_B+s_{12} r_D)\right]^{n+1}_{i,j}=\frac{\phi}{\Delta t}\left[(\rho_g S_g Y_1+\rho_w S_w)^{n+1}_{i,j}-(\rho_g S_g Y_1+\rho_w S_w)^{n}_{i,j}\right]. \quad \text{(A.1)}$$

A.2 The Discretization of Heavy Oil Mass Conservation Equation

$$\frac{6.328}{\Delta x}\left[\left(\frac{\rho_g K K_{rg} Y_2}{\mu_g}\right)^{n+1}_{i,j}\left(\frac{P_{g_{i+1,j}}-P_{g_{i,j}}}{\Delta x}\right)^{n+1}-\left(\frac{\rho_g K K_{rg} Y_2}{\mu_g}\right)^{n+1}_{i-1,j}\left(\frac{P_{g_{i,j}}-P_{g_{i-1,j}}}{\Delta x}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta x}\left[\left(\frac{\rho_g^2 K K_{rg} Y_2 M_g}{\mu_g}\right)^{n+1}_{i,j}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}-\left(\frac{\rho_g^2 K K_{rg} Y_2 M_g}{\mu_g}\right)^{n+1}_{i-1,j}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}\right]$$

$$+\frac{6.328}{\Delta x}\left[\left(\frac{\rho_o K K_{ro} X_2}{\mu_o}\right)^{n+1}_{i,j}\left(\frac{P_{o_{i+1,j}}-P_{o_{i,j}}}{\Delta x}\right)^{n+1}-\left(\frac{\rho_o K K_{ro} X_2}{\mu_o}\right)^{n+1}_{i-1,j}\left(\frac{P_{o_{i,j}}-P_{o_{i-1,j}}}{\Delta x}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta x}\left[\left(\frac{\rho_o^2 K K_{ro} X_2 M_o}{\mu_o}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}-\left(\frac{\rho_o^2 K K_{ro} X_2 M_o}{\mu_o}\right)_{i-1,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}\right]$$

$$+\frac{6.328}{\Delta y}\left[\left(\frac{\rho_g K K_{rg} Y_2}{\mu_g}\right)_{i,j}^{n+1}\left(\frac{P_{g_{i,j+1}}-P_{g_{i,j}}}{\Delta y}\right)^{n+1}-\left(\frac{\rho_g K K_{rg} Y_2}{\mu_g}\right)_{i,j-1}^{n+1}\left(\frac{P_{g_{i,j}}-P_{g_{i,j-1}}}{\Delta y}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta y}\left[\left(\frac{\rho_g^2 K K_{rg} Y_2 M_g}{\mu_g}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}-\left(\frac{\rho_g^2 K K_{rg} Y_2 M_g}{\mu_g}\right)_{i,j-1}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}\right]$$

$$+\frac{6.328}{\Delta y}\left[\left(\frac{\rho_o K K_{ro} X_2}{\mu_o}\right)_{i,j}^{n+1}\left(\frac{P_{o_{i,j+1}}-P_{o_{i,j}}}{\Delta y}\right)^{n+1}-\left(\frac{\rho_o K K_{ro} X_2}{\mu_o}\right)_{i,j-1}^{n+1}\left(\frac{P_{o_{i,j}}-P_{o_{i,j-1}}}{\Delta y}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta y}\left[\left(\frac{\rho_o^2 K K_{ro} X_2 M_o}{\mu_o}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}-\left(\frac{\rho_o^2 K K_{ro} X_2 M_o}{\mu_o}\right)_{i,j-1}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}\right]$$

$$+\left[q_2-(r_B+r_C)\right]_{i,j}^{n+1}=\frac{\phi}{\Delta t}\left[(\rho_g S_g Y_2+\rho_o S_o X_2)_{i,j}^{n+1}-(\rho_g S_g Y_2+\rho_o S_o X_2)_{i,j}^{n}\right]. \qquad \text{(A.2)}$$

A.3 The Discretization of Light Oil Mass Conservation Equation

$$\frac{6.328}{\Delta x}\left[\left(\frac{\rho_g K K_{rg} Y_3}{\mu_g}\right)_{i,j}^{n+1}\left(\frac{P_{g_{i+1,j}}-P_{g_{i,j}}}{\Delta x}\right)^{n+1}-\left(\frac{\rho_g K K_{rg} Y_3}{\mu_g}\right)_{i-1,j}^{n+1}\left(\frac{P_{g_{i,j}}-P_{g_{i-1,j}}}{\Delta x}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta x}\left[\left(\frac{\rho_g^2 K K_{rg} Y_3 M_g}{\mu_g}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}-\left(\frac{\rho_g^2 K K_{rg} Y_3 M_g}{\mu_g}\right)_{i-1,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}\right]$$

$$+\frac{6.328}{\Delta x}\left[\left(\frac{\rho_o K K_{ro} X_3}{\mu_o}\right)^{n+1}_{i,j}\left(\frac{P_{o_{i+1,j}}-P_{o_{i,j}}}{\Delta x}\right)^{n+1}-\left(\frac{\rho_o K K_{ro} X_3}{\mu_o}\right)^{n+1}_{i-1,j}\left(\frac{P_{o_{i,j}}-P_{o_{i-1,j}}}{\Delta x}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta x}\left[\left(\frac{\rho_o^2 K K_{ro} X_3 M_o}{\mu_o}\right)^{n+1}_{i,j}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}-\left(\frac{\rho_o^2 K K_{ro} X_3 M_o}{\mu_o}\right)^{n+1}_{i-1,j}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}\right]$$

$$+\frac{6.328}{\Delta y}\left[\left(\frac{\rho_g K K_{rg} Y_3}{\mu_g}\right)^{n+1}_{i,j}\left(\frac{P_{g_{i,j+1}}-P_{g_{i,j}}}{\Delta y}\right)^{n+1}-\left(\frac{\rho_g K K_{rg} Y_3}{\mu_g}\right)^{n+1}_{i,j-1}\left(\frac{P_{g_{i,j}}-P_{g_{i,j-1}}}{\Delta y}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta y}\left[\left(\frac{\rho_g^2 K K_{rg} Y_3 M_g}{\mu_g}\right)^{n+1}_{i,j}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}-\left(\frac{\rho_g^2 K K_{rg} Y_3 M_g}{\mu_g}\right)^{n+1}_{i,j-1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}\right]$$

$$+\frac{6.328}{\Delta y}\left[\left(\frac{\rho_o K K_{ro} X_3}{\mu_o}\right)^{n+1}_{i,j}\left(\frac{P_{o_{i,j+1}}-P_{o_{i,j}}}{\Delta y}\right)^{n+1}-\left(\frac{\rho_o K K_{ro} X_3}{\mu_o}\right)^{n+1}_{i,j-1}\left(\frac{P_{o_{i,j}}-P_{o_{i,j-1}}}{\Delta y}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta y}\left[\left(\frac{\rho_o^2 K K_{ro} X_3 M_o}{\mu_o}\right)^{n+1}_{i,j}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}-\left(\frac{\rho_o^2 K K_{ro} X_3 M_o}{\mu_o}\right)^{n+1}_{i,j-1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}\right]$$

$$+\left[q_3-(r_A-s_7 r_C)\right]^{n+1}_{i,j}=\frac{\phi}{\Delta t}\left[(\rho_g S_g Y_3+\rho_o S_o X_3)^{n+1}_{i,j}-(\rho_g S_g Y_3+\rho_o S_o X_3)^{n}_{i,j}\right]. \qquad \text{(A.3)}$$

A.4 The Discretization of Inert Gas Mass Conservation Equation

$$\frac{6.328}{\Delta x}\left[\left(\frac{\rho_g K K_{rg} Y_4}{\mu_g}\right)^{n+1}_{i,j}\left(\frac{P_{g_{i+1,j}}-P_{g_{i,j}}}{\Delta x}\right)^{n+1}-\left(\frac{\rho_g K K_{rg} Y_4}{\mu_g}\right)^{n+1}_{i-1,j}\left(\frac{P_{g_{i,j}}-P_{g_{i-1,j}}}{\Delta x}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta x}\left[\left(\frac{\rho_g^2 K K_{rg} Y_4 M_g}{\mu_g}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathrm{g}}{g_c}\frac{\partial D}{\partial x}-\left(\frac{\rho_g^2 K K_{rg} Y_4 M_g}{\mu_g}\right)_{i-1,j}^{n+1}\frac{1}{144}\frac{\mathrm{g}}{g_c}\frac{\partial D}{\partial x}\right]$$

$$+\frac{6.328}{\Delta y}\left[\left(\frac{\rho_g K K_{rg} Y_4}{\mu_g}\right)_{i,j}^{n+1}\left(\frac{P_{g_{i,j+1}}-P_{g_{i,j}}}{\Delta y}\right)^{n+1}-\left(\frac{\rho_g K K_{rg} Y_4}{\mu_g}\right)_{i,j-1}^{n+1}\left(\frac{P_{g_{i,j}}-P_{g_{i,j-1}}}{\Delta y}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta y}\left[\left(\frac{\rho_g^2 K K_{rg} Y_4 M_g}{\mu_g}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathrm{g}}{g_c}\frac{\partial D}{\partial y}-\left(\frac{\rho_g^2 K K_{rg} Y_4 M_g}{\mu_g}\right)_{i,j-1}^{n+1}\frac{1}{144}\frac{\mathrm{g}}{g_c}\frac{\partial D}{\partial y}\right]$$

$$+\left[q_4+(s_2 r_A+s_5 r_B+s_9 r_C+s_{11} r_D)\right]_{i,j}^{n+1}=\frac{\phi}{\Delta t}\left[(\rho_g S_g Y_4)_{i,j}^{n+1}-(\rho_g S_g Y_4)_{i,j}^{n}\right]. \qquad \text{(A.4)}$$

A.5 The Discretization of Oxygen Mass Conservation Equation

$$\frac{6.328}{\Delta x}\left[\left(\frac{\rho_g K K_{rg} Y_5}{\mu_g}\right)_{i,j}^{n+1}\left(\frac{P_{g_{i+1,j}}-P_{g_{i,j}}}{\Delta x}\right)^{n+1}-\left(\frac{\rho_g K K_{rg} Y_5}{\mu_g}\right)_{i-1,j}^{n+1}\left(\frac{P_{g_{i,j}}-P_{g_{i-1,j}}}{\Delta x}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta x}\left[\left(\frac{\rho_g^2 K K_{rg} Y_5 M_g}{\mu_g}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathrm{g}}{g_c}\frac{\partial D}{\partial x}-\left(\frac{\rho_g^2 K K_{rg} Y_5 M_g}{\mu_g}\right)_{i-1,j}^{n+1}\frac{1}{144}\frac{\mathrm{g}}{g_c}\frac{\partial D}{\partial x}\right]$$

$$+\frac{6.328}{\Delta y}\left[\left(\frac{\rho_g K K_{rg} Y_5}{\mu_g}\right)_{i,j}^{n+1}\left(\frac{P_{g_{i,j+1}}-P_{g_{i,j}}}{\Delta y}\right)^{n+1}-\left(\frac{\rho_g K K_{rg} Y_5}{\mu_g}\right)_{i,j-1}^{n+1}\left(\frac{P_{g_{i,j}}-P_{g_{i,j-1}}}{\Delta y}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta y}\left[\left(\frac{\rho_g^2 K K_{rg} Y_5 M_g}{\mu_g}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathrm{g}}{g_c}\frac{\partial D}{\partial y}-\left(\frac{\rho_g^2 K K_{rg} Y_5 M_g}{\mu_g}\right)_{i,j-1}^{n+1}\frac{1}{144}\frac{\mathrm{g}}{g_c}\frac{\partial D}{\partial y}\right]$$

$$+\left[q_5-(s_1r_A+s_4r_B+s_{10}r_D)\right]_{i,j}^{n+1}=\frac{\phi}{\Delta t}\left[(\rho_gS_gY_5)_{i,j}^{n+1}-(\rho_gS_gY_5)_{i,j}^{n}\right]. \qquad \text{(A.5)}$$

A.6 The Discretization of Coke Conservation Equation

$$(s_8r_C-r_D)^{n+1}=\frac{1}{\Delta t}\left(C_c^{n+1}-C_c^{n}\right). \qquad \text{(A.6)}$$

A.7 The Discretization of Energy Conservation Equation

$$\frac{\lambda}{\Delta x^2}(T_{i-1,j}-2T_{i,j}+T_{i+1,j})^{n+1}$$

$$+\frac{6.328}{\Delta x}\left[\left(\frac{\rho_gKK_{rg}h_g}{\mu_g}\right)_{i,j}^{n+1}\left(\frac{P_{g_{i+1,j}}-P_{g_{i,j}}}{\Delta x}\right)^{n+1}-\left(\frac{\rho_gKK_{rg}h_g}{\mu_g}\right)_{i-1,j}^{n+1}\left(\frac{P_{g_{i,j}}-P_{g_{i-1,j}}}{\Delta x}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta x}\left[\left(\frac{\rho_g^2KK_{rg}h_gM_g}{\mu_g}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}-\left(\frac{\rho_g^2KK_{rg}h_gM_g}{\mu_g}\right)_{i-1,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}\right]$$

$$+\frac{6.328}{\Delta x}\left[\left(\frac{\rho_oKK_{ro}h_o}{\mu_o}\right)_{i,j}^{n+1}\left(\frac{P_{o_{i+1,j}}-P_{o_{i,j}}}{\Delta x}\right)^{n+1}-\left(\frac{\rho_oKK_{ro}h_o}{\mu_o}\right)_{i-1,j}^{n+1}\left(\frac{P_{o_{i,j}}-P_{o_{i-1,j}}}{\Delta x}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta x}\left[\left(\frac{\rho_o^2KK_{ro}h_oM_o}{\mu_o}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}-\left(\frac{\rho_o^2KK_{ro}h_oM_o}{\mu_o}\right)_{i-1,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}\right]$$

$$+\frac{6.328}{\Delta x}\left[\left(\frac{\rho_w K K_{rw} h_w}{\mu_w}\right)_{i,j}^{n+1}\left(\frac{P_{w_{i+1,j}}-P_{w_{i,j}}}{\Delta x}\right)^{n+1}-\left(\frac{\rho_w K K_{rw} h_w}{\mu_w}\right)_{i-1,j}^{n+1}\left(\frac{P_{w_{i,j}}-P_{w_{i-1,j}}}{\Delta x}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta x}\left[\left(\frac{\rho_w^2 K K_{rw} h_w M_w}{\mu_w}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}-\left(\frac{\rho_w^2 K K_{rw} h_w M_w}{\mu_w}\right)_{i-1,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial x}\right]$$

$$+\frac{\lambda}{\Delta y^2}(T_{i,j-1}-2T_{i,j}+T_{i,j+1})^{n+1}$$

$$+\frac{6.328}{\Delta y}\left[\left(\frac{\rho_g K K_{rg} h_g}{\mu_g}\right)_{i,j}^{n+1}\left(\frac{P_{g_{i,j+1}}-P_{g_{i,j}}}{\Delta y}\right)^{n+1}-\left(\frac{\rho_g K K_{rg} h_g}{\mu_g}\right)_{i,j-1}^{n+1}\left(\frac{P_{g_{i,j}}-P_{g_{i,j-1}}}{\Delta y}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta y}\left[\left(\frac{\rho_g^2 K K_{rg} h_g M_g}{\mu_g}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}-\left(\frac{\rho_g^2 K K_{rg} h_g M_g}{\mu_g}\right)_{i,j-1}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}\right]$$

$$+\frac{6.328}{\Delta y}\left[\left(\frac{\rho_o K K_{ro} h_o}{\mu_o}\right)_{i,j}^{n+1}\left(\frac{P_{o_{i,j+1}}-P_{o_{i,j}}}{\Delta y}\right)^{n+1}-\left(\frac{\rho_o K K_{ro} h_o}{\mu_o}\right)_{i,j-1}^{n+1}\left(\frac{P_{o_{i,j}}-P_{o_{i,j-1}}}{\Delta y}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta y}\left[\left(\frac{\rho_o^2 K K_{ro} h_o M_o}{\mu_o}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}-\left(\frac{\rho_o^2 K K_{ro} h_o M_o}{\mu_o}\right)_{i,j-1}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}\right]$$

$$+\frac{6.328}{\Delta y}\left[\left(\frac{\rho_w K K_{rw} h_w}{\mu_w}\right)_{i,j}^{n+1}\left(\frac{P_{w_{i,j+1}}-P_{w_{i,j}}}{\Delta y}\right)^{n+1}-\left(\frac{\rho_w K K_{rw} h_w}{\mu_w}\right)_{i,j-1}^{n+1}\left(\frac{P_{w_{i,j}}-P_{w_{i,j-1}}}{\Delta y}\right)^{n+1}\right]$$

$$-\frac{6.328}{\Delta y}\left[\left(\frac{\rho_w^2 K K_{rw} h_w M_w}{\mu_w}\right)_{i,j}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}-\left(\frac{\rho_w^2 K K_{rw} h_w M_w}{\mu_w}\right)_{i,j-1}^{n+1}\frac{1}{144}\frac{\mathbf{g}}{g_c}\frac{\partial D}{\partial y}\right]$$

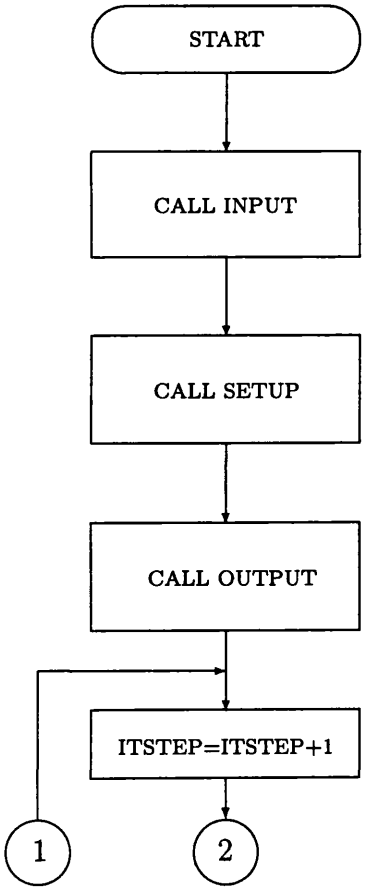
$$+[(q_g h^*{}_g+q_o h^*{}_o+q_w h^*{}_w)+(H_A r_A+H_B r_B+H_C r_C+H_D r_D)+H_{inj}-H_{loss}]_{i,j}^{n+1}$$

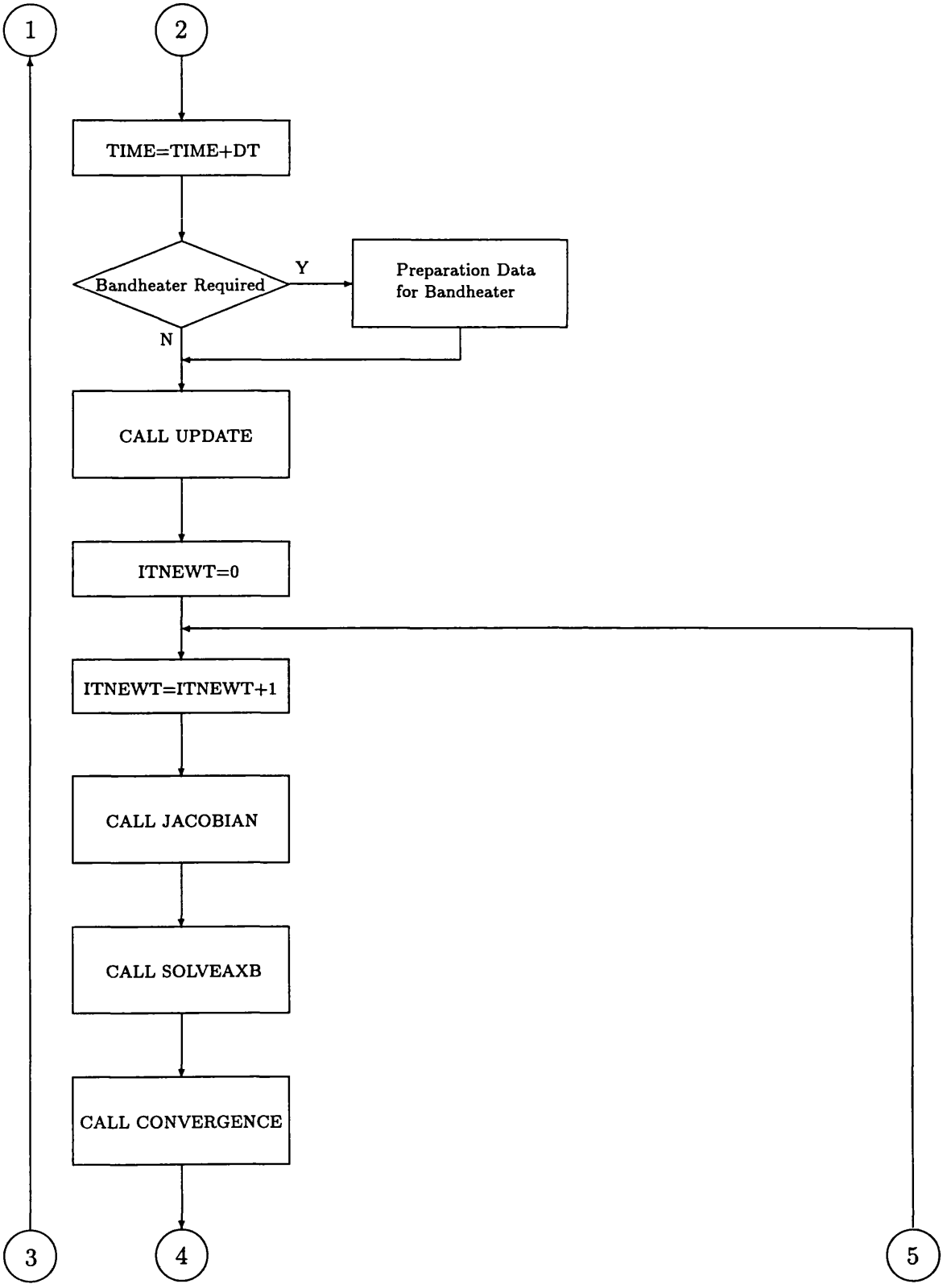
$$= \frac{1}{\Delta t}\left[(C_c U_c)_{i,j}^{n+1} - (C_c U_c)_{i,j}^{n}\right] + \frac{(1-\phi)}{\Delta t}\left[(\rho_r U_r)_{i,j}^{n+1} - (\rho_r U_r)_{i,j}^{n}\right] + \frac{\phi}{\Delta t}\left[(\rho_g S_g U_g)_{i,j}^{n+1} - (\rho_g S_g U_g)_{i,j}^{n}\right]$$

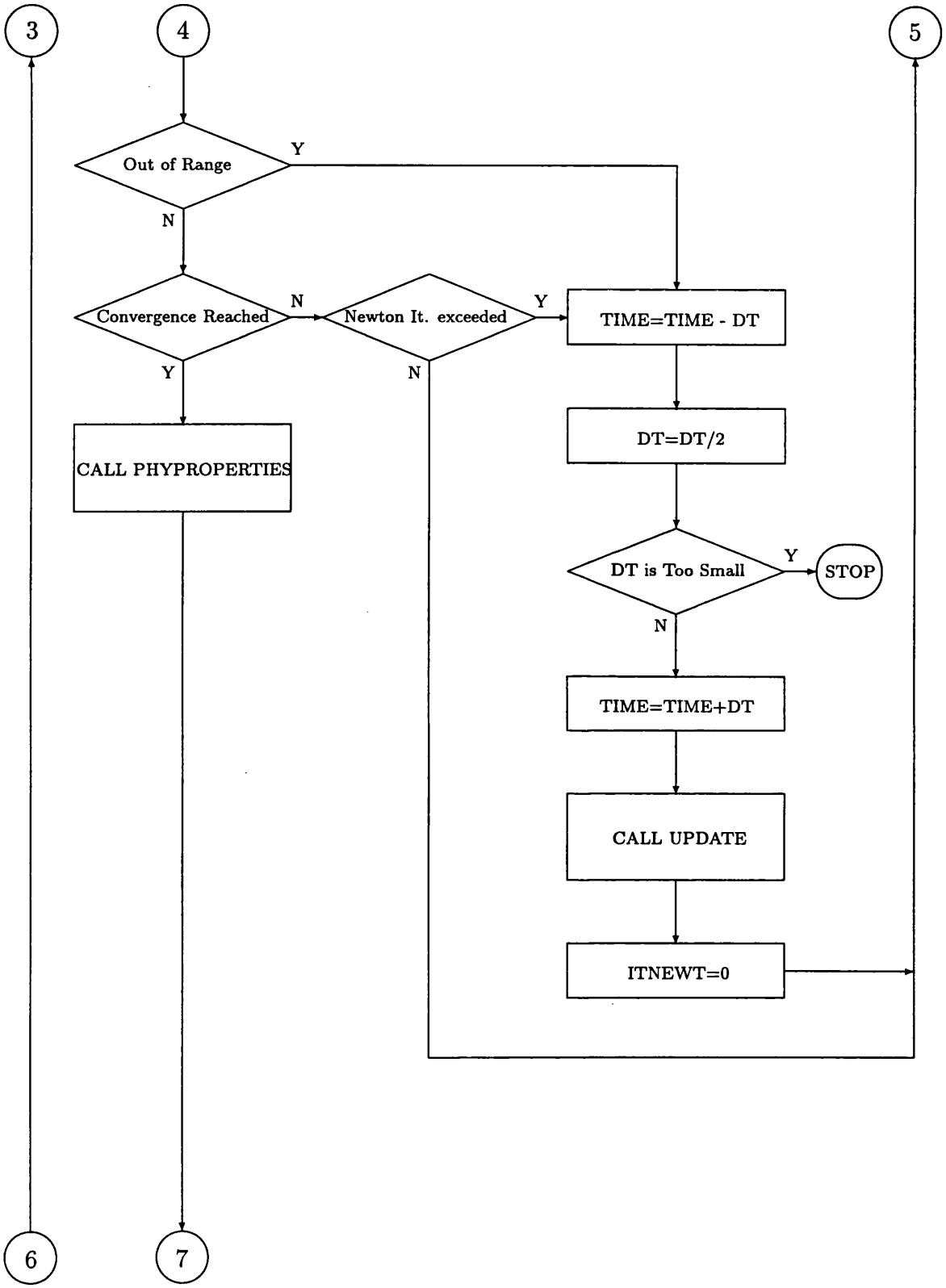
$$+ \frac{\phi}{\Delta t}\left[(\rho_o S_o U_o)_{i,j}^{n+1} - (\rho_o S_o U_o)_{i,j}^{n}\right] + \frac{\phi}{\Delta t}\left[(\rho_w S_w U_w)_{i,j}^{n+1} - (\rho_w S_w U_w)_{i,j}^{n}\right]. \qquad \text{(A.7)}$$

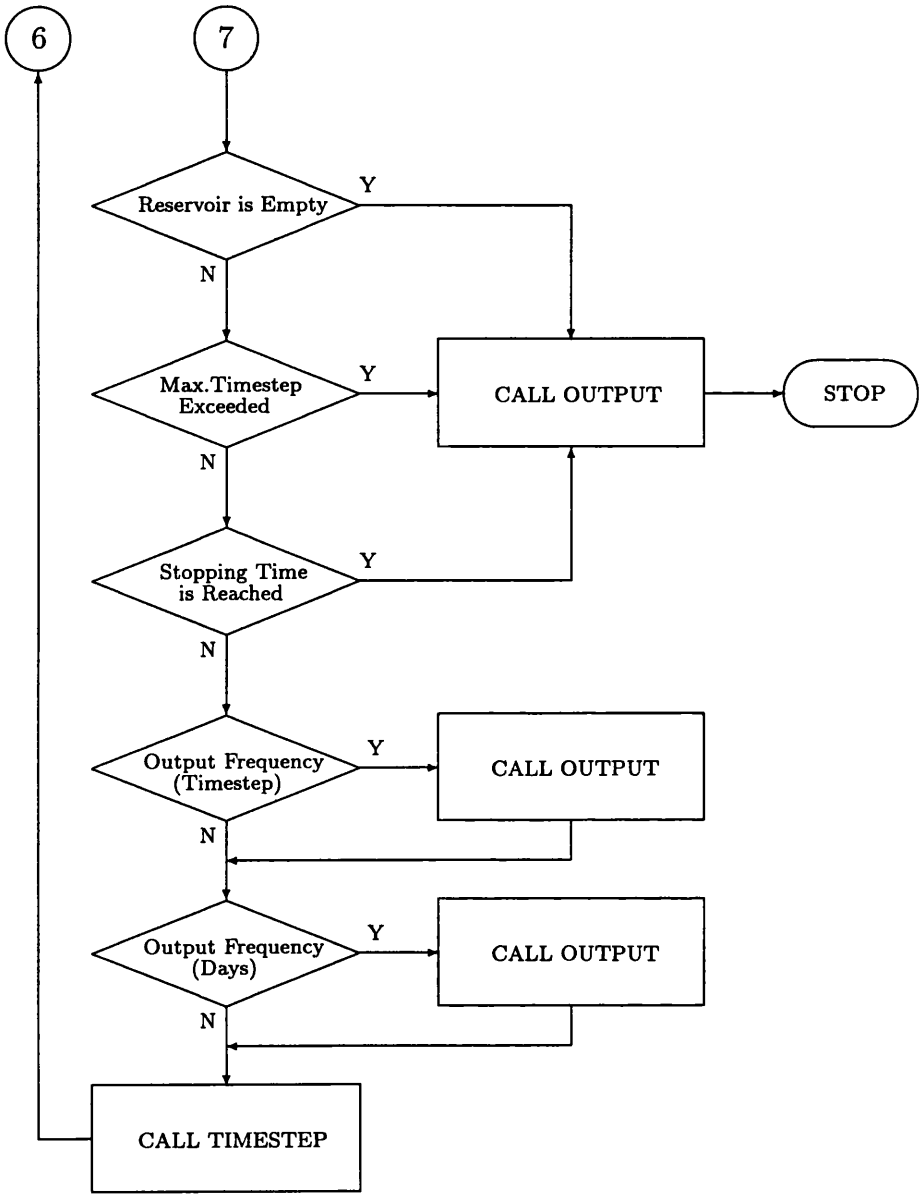
Appendix B

Flowchart of The In-Situ Combustion Simulator




1
2
TIME=TIME+DT
Bandheater Required
Y
Preparation Data
for Bandheater
N
CALL UPDATE
ITNEWT=0
ITNEWT=ITNEWT+1
CALL JACOBIAN
CALL SOLVEAXB
CALL CONVERGENCE
3
4
5


3
4
5
Out of Range
Y
N
Convergence Reached
N
Newton It. exceeded
Y
N
Y
TIME=TIME - DT
DT=DT/2
CALL PHYPROPERTIES
DT is Too Small
Y
STOP
N
TIME=TIME+DT
CALL UPDATE
ITNEWT=0
6
7


6
7
Reservoir is Empty
Y
N
Max.Timestep Exceeded
Y
CALL OUTPUT
STOP
N
Stopping Time is Reached
Y
N
Output Frequency (Timestep)
Y
CALL OUTPUT
N
Output Frequency (Days)
Y
CALL OUTPUT
N
CALL TIMESTEP

Appendix C

Listings of The Computer Program

```
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      PROGRAM BUISCOM
      INCLUDE 'DECLARATION'
      INTEGER I,J,NEWTOT,ITTOT
C
      CALL INPUT
      ZM = OUTFQD - 1.0
      ZP = OUTFQD + 1.0
      CALL SETUP
      CALL OUTPUT
      ITTOT = 0
      NEWTOT = 0
      ICITE = 0
      IFLCOK = 0
10    ITSTEP = ITSTEP + 1
      TIME = TIME + DT
C
C**** If preheat required then
C
      IF(PRHT .GT. 0.0 .AND. TIME .GE. PRHT)THEN
        QINJ1(1,1) = QHOLD1
        QINJ2(1,1) = QHOLD2
        QINJ3(1,1) = QHOLD3
        QINJ4(1,1) = QHOLD4
```

```
        QINJ5(1,1) = QHOLD5
      ENDIF
C
      CALL UPDATE
      ITNEWT = 0
      ICITE = 0
20    ITNEWT = ITNEWT + 1
      CALL JACOBIAN
      CALL SOLVEAXB
      CALL CONVERGENCE
      IF(IFLCF1 .EQ. 1)THEN
        WRITE(11,305)ITSTEP,DT,TIME,ITNEWT,ICITE,ITTOT,NEWTOT
        GOTO 30
      ENDIF
      IF(IFLCF2 .EQ. 1)THEN
        IF(ITNEWT .EQ. MXIT)THEN
          WRITE(11,315)ITSTEP,DT,TIME,ITNEWT,ICITE,ITTOT,NEWTOT
          GOTO 30
        ENDIF
        GOTO 20
      ENDIF
      GOTO 40
30    TIME = TIME - DT
      DT = DT/2.0
      IF(DT .LT. 1.0D-06)THEN
        WRITE(11,320)ITSTEP,DT,TIME,ITNEWT,ICITE,ITTOT,NEWTOT
        STOP
      ENDIF
      TIME = TIME + DT
      CALL RESET
      ITNEWT = 0
      ICITE = 0
      GOTO 20
40    CONTINUE
      DO 32 J=1,NGBY
        DO 33 I=1,NGBX
          CALL PHYPROPERTIES(I,J)
          IF(IFLCOK .EQ. 1) GOTO 771
33      CONTINUE
32    CONTINUE
      GOTO 772
771   CONTINUE
      WRITE(11,785)ITSTEP,DT,TIME,ITNEWT,ICITE,ITTOT,NEWTOT
      TIME = TIME - DT
      DT = DT/2.0
```

```
      IF(DT .LT. 1.0D-06)THEN
        WRITE(11,320)ITSTEP,DT,TIME,ITNEWT,ICITE,ITTOT,NEWTOT
        STOP
      ENDIF
      TIME = TIME + DT
      CALL RESET
      ITNEWT = 0
      ICITE = 0
      IFLCOK = 0
      GOTO 20
772   CONTINUE
      NEWTOT = NEWTOT + ITNEWT
      ITTOT = ITTOT + ICITE
      WRITE(11,330)ITSTEP,DT,TIME,ITNEWT,ICITE,ITTOT,NEWTOT
      DTOLD = DT
      IF(SO(NGBX,NGBY) .LT. 1.0D-04) THEN
        WRITE(11,303) TIME,ITSTEP
        WRITE(31,303) TIME,ITSTEP
        GOTO 50
      ENDIF
      IF(ITSTEP .GE. MSTEP) THEN
        WRITE(11,301) TIME,ITSTEP
        WRITE(31,301) TIME,ITSTEP
        GOTO 50
      ENDIF
      IF(TIME .GE. ETIM) THEN
        WRITE(11,302) TIME,ITSTEP
        WRITE(31,302) TIME,ITSTEP
        GOTO 50
      ENDIF
      IF((ITSTEP/IOUTFQ)*IOUTFQ .EQ. ITSTEP) CALL OUTPUT
      IF(TIME .GT. ZM .AND. TIME .LE. ZP) THEN
        CALL OUTPUT
        ZO = ZO + OUTFQD
        ZM = ZO - 1.0
        ZP = ZO + 1.0
      ENDIF
      CALL TIMESTEP
      GOTO 10
50    CONTINUE
      CALL OUTPUT
      WRITE(11,304)
      WRITE(31,304)
C
305   FORMAT(I4,4X,F10.8,3X,F8.4,4X,I2,4X,I4,4X,I7,4X,I6,4X,'RESET')
```

```
315   FORMAT(I4,4X,F10.8,3X,F8.4,4X,I2,4X,I4,4X,I7,4X,I6,11X,'RESET')
320   FORMAT(I4,4X,F10.8,3X,F8.4,4X,I2,4X,I4,4X,I7,4X,I6,'DT too small')
785   FORMAT(I4,4X,F10.8,3X,F8.4,4X,I2,4X,I4,4X,I7,4X,I6,7X,'RESET')
330   FORMAT(I4,4X,F10.8,3X,F8.4,4X,I2,4X,I4,4X,I7,4X,I6)
303   FORMAT(//16X,40('*')/26X,'NOW RESERVOIR IS EMPTY ',/
     +19X,'THE TIME',8X,'=',F13.6,2X,'days'/
     +19X,'TIMESTEP NO.',4X,'=',I13,/16X,40('*'),///)
301   FORMAT(//16X,40('*')/
     +16X,'STOPPING MAX. NUMBER OF TIMESTEP REACHED'//
     +19X,'THE TIME',8X,'=',F11.6,2X,'days'/
     +19X,'TIMESTEP NO.',4X,'=',I11,/16X,40('*'))
302   FORMAT(//16X,40('*')/23X,'STOPPING END TIME REACHED'//
     +19X,'THE TIME',8X,'=',F11.6,2X,'days'/
     +19X,'TIMESTEP NO.',4X,'=',I11,/16X,40('*'))
304   FORMAT(///16X,40('*')/16X,'*',38X,'*'/17('*'),6X,
     +'THE END OF THE OUTPUT FILE',6X,17('*')/
     +16X,'*',38X,'*'/16X,40('*')///)
C
      STOP
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE INPUT
      INCLUDE 'DECLARATION'
      INTEGER I,J
C
      CHARACTER*72 COMMENT
      OPEN (UNIT=1,FILE='INPUT.DAT',STATUS='OLD')
C
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,*) IRUNNO
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
```

```
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)RLEN,RWID,DZ,NGBX,NGBY
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)SLOPEX,SLOPEY,G
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)((PRESG(I,J),I=1,NGBX),J=NGBY,1,-1)
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)((SW(I,J),I=1,NGBX),J=NGBY,1,-1)
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)((SO(I,J),I=1,NGBX),J=NGBY,1,-1)
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)((TEMP(I,J),I=1,NGBX),J=NGBY,1,-1)
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)((Y5(I,J),I=1,NGBX),J=NGBY,1,-1)
READ(1,'(A72)')COMMENT
```

```
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)((X2(I,J),I=1,NGBX),J=NGBY,1,-1)
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)((CC(I,J),I=1,NGBX),J=NGBY,1,-1)
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)((BHTEMP(I,J),I=1,NGBX),J=NGBY,1,-1)
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)PRHT,BTHCD
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)PERM,THCD,THCDCAP,GASCST
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)PORE,TREF,PREF,PINDEX,PRODP
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)COKMAX,IHLS
```

```
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)QINJ1(1,1),QINJ2(1,1),QINJ3(1,1),QINJ4(1,1),QINJ5(1,1)
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)TINJ
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)CPG1,CPG2,CPW1,CPW2
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)EQUW2,EQUW1,EQUW3
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)EQUHO1,EQUHO2,EQUHO3
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
```

```
READ(1,'(A72)')COMMENT
READ(1,*)EQUL01,EQUL02,EQUL03
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)DW1,DW2,DW3
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)DH01,DH02,DH03,DL01,DL02,DL03
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)VG15,VG14,VG13,VG12,VG11
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)VG25,VG24,VG23,VG22,VG21
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)VW1,VW2,VW3,VW4
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
```

```
READ(1,'(A72)')COMMENT
READ(1,*)VLO1,VLO2,VHO1,VHO2
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)WTML1,WTML2,WTML3,WTML4,WTML5,WTML6
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)AKRWRO,AKRGRO,AKROCW,SWC,SORW,SORG,SGC,ZW,ZG,ZOG,ZOW
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)CWA,CHO,CLO,CIG,CO2
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)EHO1,TCLO,TCHO,EW1,TCWT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)CCK,CRK
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
```

```
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)ARHA,ARHB,ARHC,ARHD
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)EENA,EENB,EENC,EEND
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)HRA,HRB,HRC,HRD
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)DTINIT,DTMAX,ETIM
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,*)URMX,MXIT,MSTEP
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
READ(1,'(A72)')COMMENT
```

```
      READ(1,'(A72)')COMMENT
      READ(1,*)IOUTFQ,OUTFQD
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,*)PNORM,SWNORM,SONORM,TNORM,YNORM,XNORM
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,*)PCON,SWCON,SOCON,TCON,YCON,XCON
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,*)ITMET,MGMRES,ILUF
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,*)ISCAL,MAXITE,TOL
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,'(A72)')COMMENT
      READ(1,*)IOUT3
      READ(1,'(A72)')COMMENT
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
```

```
      SUBROUTINE SETUP
      INCLUDE 'DECLARATION'
      INTEGER I,J
C
      ITSTEP = 0
      IFLCF1 = 0
      IFLCF2 = 0
      N = NGBX*NGBY*6
      OPEN(UNIT=11,FILE='OUTPUT.N',STATUS='NEW')
      IF(IOUT3 .EQ. 1) THEN
        OPEN(UNIT=31,FILE='OUTPUT.L',STATUS='NEW')
      ELSE
        OPEN(UNIT=31,FILE='OUTPUT.S',STATUS='NEW')
      ENDIF
      TIME = 0.0
      DT = DTINIT
      DTOLD = 0.0
      G = G/32.174
      DX = RLEN/NGBX
      DDDX = DSIN(SLOPEX*3.14159/180.0)*DX
      DY = RWID/NGBY
      DDDY = DSIN(SLOPEY*3.14159/180.0)*DY
      VOL = DX*DY*DZ
      TINJ = TINJ + 459.69
      DO 10 J=1,NGBY
        DO 11 I=1,NGBX
          TEMP(I,J) = TEMP(I,J) + 459.69
          BHTEMP(I,J) = BHTEMP(I,J) + 459.69
          HLOSS(I,J) = 0.0
          TINIT(I,J) = TEMP(I,J)
          DHL = 0.0
          PHL(I,J) = 0.0
          QHL(I,J) = 0.0
          ENCAP(I,J) = 0.0
          PTDHL = 0.0
          PTPHL(I,J) = 0.0
          PTQHL(I,J) = 0.0
          PTENCAP(I,J) = 0.0
          RA(I,J) = 0.0
          RB(I,J) = 0.0
          RC(I,J) = 0.0
          RD(I,J) = 0.0
11      CONTINUE
10    CONTINUE
      QINJ1(1,1) = QINJ1(1,1)/VOL
```

```
      QINJ2(1,1) = QINJ2(1,1)/VOL
      QINJ3(1,1) = QINJ3(1,1)/VOL
      QINJ4(1,1) = QINJ4(1,1)/VOL
      QINJ5(1,1) = QINJ5(1,1)/VOL
      IF(PRHT .GT. 0.0) THEN
        QHOLD1 = QINJ1(1,1)
        QINJ1(1,1) = 0.0
        QHOLD2 = QINJ2(1,1)
        QINJ2(1,1) = 0.0
        QHOLD3 = QINJ3(1,1)
        QINJ3(1,1) = 0.0
        QHOLD4 = QINJ4(1,1)
        QINJ4(1,1) = 0.0
        QHOLD5 = QINJ5(1,1)
        QINJ5(1,1) = 0.0
      ENDIF
      DO 34 J = 2,NGBY
        DO 35 I = 2,NGBX
          QINJ1(I,J) = 0.0000
          QINJ2(I,J) = 0.0000
          QINJ3(I,J) = 0.0000
          QINJ4(I,J) = 0.0000
          QINJ5(I,J) = 0.0000
35      CONTINUE
34    CONTINUE
      DO 20 J=1,NGBY
        DO 21 I=1,NGBX
          CALL PHYPROPERTIES(I,J)
21      CONTINUE
20    CONTINUE
      CALL UPDATE
      DO 5 I=1,N
        DO 6 J=1,N
          A(I,J) = 0.0
6       CONTINUE
5     CONTINUE
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE PHYPROPERTIES(I,J)
      INCLUDE 'DECLARATION'
      INTEGER I,J
```

```
C
      SG(I,J) = 1.00 - (SO(I,J) + SW(I,J))
      IF(SG(I,J) .LT. 0.000) THEN
        SG(I,J) = 0.000
      ENDIF
      X3(I,J) = 1.00 - X2(I,J)
C
C**** Capillary pressures
C
      PCGO = CPG1*SG(I,J) + CPG2
      PCOW = CPW1*SW(I,J) + CPW2
      PRESO(I,J) = PRESG(I,J) - PCGO
      PRESW(I,J) = PRESO(I,J) - PCOW
C
C**** Equilibrium ratios
C
      EK1(I,J) = (1.0/PRESG(I,J))*((TEMP(I,J)- EQUW1)/EQUW2)**EQUW3
      EK1(I,J) = EK1(I,J)*(SW(I,J)/(SW(I,J)+1.0D-3))
      T1 = (1.0/PRESG(I,J))*(SO(I,J)/(SO(I,J)+1.0D-3))
      EK2(I,J) = T1*DEXP(EQUHO1+EQUHO2/(TEMP(I,J)-EQUHO3))
      EK3(I,J) = (1.0/PRESG(I,J))*DEXP(EQULO1+EQULO2/(TEMP(I,J)-EQULO3))
C
C**** Gas phase mole fractions
C
      Y1(I,J) = EK1(I,J)
      Y2(I,J) = EK2(I,J)*X2(I,J)
      Y3(I,J) = EK3(I,J)*X3(I,J)
      Y4(I,J) = 1.00 - Y1(I,J) - Y2(I,J) - Y3(I,J) - Y5(I,J)
C
C**** Molecular weights
C
      WTMLG(I,J) = WTML1*Y1(I,J)+WTML2*Y2(I,J)+WTML3*Y3(I,J)+
     *             WTML4*Y4(I,J)+WTML5*Y5(I,J)
      WTMLO(I,J) = WTML2*X2(I,J)+WTML3*X3(I,J)
      WTMLW(I,J) = WTML1
C
C**** Densities
C
      DENW(I,J) =DW1*(1.0+DW2*(PRESW(I,J)-PREF)-DW3*(TEMP(I,J)-TREF))
      DTEM2 = 1.0 + DHO1*(TEMP(I,J)-TREF)
      DPRE2 = 1.0 - DHO2*(PRESO(I,J)-PREF)
      DENO2 = DHO3*DTEM2*DPRE2
      DTEM3 = 1.0 + DLO1*(TEMP(I,J)-TREF)
      DPRE3 = 1.0 - DLO2*(PRESO(I,J)-PREF)
      DENO3 = DLO3*DTEM3*DPRE3
```

```
      DENO(I,J) = 1.0/(X2(I,J)*DENO2 + X3(I,J)*DENO3)
      DENG(I,J) = PRESG(I,J)/(10.73*TEMP(I,J))
C
C**** Viscosities
C
      TC = TEMP(I,J)*(5.0/9.0) - 273.2
      VISW(I,J)= VW1/(VW2 + VW3*TC + VW4*TC**2)
      VISCO2 = VHO1*DEXP(VHO2/TEMP(I,J))
      VISCO3 = VLO1*DEXP(VLO2/TEMP(I,J))
      VISO(I,J) = (VISCO2**X2(I,J))*(VISCO3**X3(I,J))
      VISCG1 = VG11*(TEMP(I,J)**VG21)
      VISCG2 = VG12*(TEMP(I,J)**VG22)
      VISCG3 = VG13*(TEMP(I,J)**VG23)
      VISCG4 = VG14*(TEMP(I,J)**VG24)
      VISCG5 = VG15*(TEMP(I,J)**VG25)
      DV3 = Y1(I,J)*VISCG1+Y2(I,J)*VISCG2+Y3(I,J)*VISCG3
      DV4 = Y4(I,J)*VISCG4 + Y5(I,J)*VISCG5
      VISG(I,J) = (DV3 + DV4)
C
C**** Relative permeabilities
C
      IF(SW(I,J) .GT. SWC) THEN
        AKRW(I,J) = AKRWRO*((SW(I,J)-SWC)/(1.0-SORW-SWC))**ZW
      ELSE
        AKRW(I,J) = 0.0
      ENDIF
      IF(SG(I,J) .GT. SGC) THEN
        AKRG(I,J) = AKRGRO*((SG(I,J)-SGC)/(1.0-SWC-SORG-SGC))**ZG
      ELSE
        AKRG(I,J) = 0.0
      ENDIF
      IF(SO(I,J) .LT. SORW) THEN
        AKRO(I,J) = 0.0
      ELSE
        DU1 = 1.0 - SWC - SORG - SGC
        DU2 = 1.0 - SWC - SORW
        AKROW = 1.0*(1.0 - ((SW(I,J) - SWC)/DU2))**ZOW
        AKROG = 1.0*(1.0 - ((SG(I,J) - SGC)/DU1))**ZOG
        IF(AKROW .GT. 1.00) AKROW = 1.00
        IF(AKROW .LT. 0.00) AKROW = 0.00
        IF(AKROG .GT. 1.00) AKROG = 1.00
        IF(AKROG .LT. 0.00) AKROG = 0.00
        AKRO(I,J) = AKROCW*((AKROW/1.0+AKRW(I,J))*(AKROG/1.0+AKRG(I,J))
     * -AKRW(I,J) - AKRG(I,J))
      ENDIF
```

```
C
C**** Enthalpies
C
      DUHG = CWA*Y1(I,J)+CHO*Y2(I,J)+CLO*Y3(I,J)+CIG*Y4(I,J)+CO2*Y5(I,J)
      HG(I,J) = DUHG*(TEMP(I,J) - TREF)
      IF(TCWT .GT. TEMP(I,J)) THEN
        HVWT = EW1*(TCWT - TEMP(I,J))**0.38
      ELSE
        HVWT = 0.00
      ENDIF
      HWG = CWA*(TEMP(I,J) - TREF)
      HW(I,J) = HWG - HVWT
      HO2 = CHO*(TEMP(I,J) - TREF)
      HO3 = CLO*(TEMP(I,J) - TREF)
      HOIL = X2(I,J)*HO2 + X3(I,J)*HO3
      TCO = X2(I,J)*TCHO + X3(I,J)*TCLO
      IF(TCO .GT. TEMP(I,J)) THEN
        HVOL = EHO1*(TCO - TEMP(I,J))**0.38
      ELSE
        HVOL = 0.00
      ENDIF
      HO(I,J) = HOIL - HVOL
      HC(I,J) = CCK*(TEMP(I,J) - TREF)
C
C**** Internal energies
C
      UG(I,J) = HG(I,J) - GASCST*TEMP(I,J)
      UO(I,J) = HO(I,J) - (1.9872/10.73)*(PRESO(I,J)/DENO(I,J))
      UW(I,J) = HW(I,J) - (1.9872/10.73)*(PRESW(I,J)/DENW(I,J))
      UR(I,J) = CRK*(TEMP(I,J) - TREF)
C
      IF(ITSTEP .GT. 0) THEN
C
C**** Coke equation
C
        CALL EQUATIONCOKE(I,J)
C
C**** Reaction Rates
C
C**** rA  :  (lo) + s1 (O2)  -->  s2 (COx)  +  s3 (H2O)
C
        DRA = ARHA * DEXP(-EENA/(GASCST*TEMP(I,J)))*Y5(I,J)*PRESG(I,J)
        RA(I,J) = DRA*PORE*DENO(I,J)*SO(I,J)*X3(I,J)
C
C**** rB  :  (ho) + s4 (O2)  -->  s5 (COx)  +  s6 (H2O)
```

```
C
        DRB = ARHB * DEXP(-EENB/(GASCST*TEMP(I,J)))*Y5(I,J)*PRESG(I,J)
        RB(I,J) = DRB*PORE*DENO(I,J)*SO(I,J)*X2(I,J)
C
C**** rC  :  (ho)           -->  s7 (lo)  +  s8 (coke)  + s9(in.g)
C
        DRC1 = ARHC * DEXP(-EENC/(GASCST*TEMP(I,J)))
        DRC2 = DRC1*PORE*DENO(I,J)*SO(I,J)*X2(I,J)
        RC(I,J) = DRC2*(1.0-(CC(I,J)/COKMAX)**5.0)
C
C**** rD  : (coke)+ s10(O2)  -->  s11 (O2) + s12 (H2O)
C
        DRD = ARHD * DEXP(-EEND/(GASCST*TEMP(I,J)))*Y5(I,J)*PRESG(I,J)
        RD(I,J) = DRD*CC(I,J)
      ENDIF
C
C**** Heat loss
C
      IF(IHLS .EQ. 1 .AND. ITSTEP .GT. 0) CALL HEATLOSS(I,J)
C
C**** Production rates
C
      IF(I .EQ. NGBX .AND. J .EQ. NGBY) THEN
        IF(ITSTEP .GT. 0) THEN
          DPG = 6.328*PINDEX*PERM*((PRESG(NGBX,NGBY) - PRODP)/(DX*DY))
          DPO = 6.328*PINDEX*PERM*((PRESO(NGBX,NGBY) - PRODP)/(DX*DY))
          DPW = 6.328*PINDEX*PERM*((PRESW(NGBX,NGBY) - PRODP)/(DX*DY))
          QG = DPG*AKRG(NGBX,NGBY)*DENG(NGBX,NGBY)/VISG(NGBX,NGBY)
          QO = DPO*AKRO(NGBX,NGBY)*DENO(NGBX,NGBY)/VISO(NGBX,NGBY)
          QW = DPW*AKRW(NGBX,NGBY)*DENW(NGBX,NGBY)/VISW(NGBX,NGBY)
          QPRO1(NGBX,NGBY) = QW
          QP1G(NGBX,NGBY) = QG*Y1(NGBX,NGBY)
          QPRO2(NGBX,NGBY) = QO*X2(NGBX,NGBY)
          QP2G(NGBX,NGBY) = QG*Y2(NGBX,NGBY)
          QPRO3(NGBX,NGBY) = QO*X3(NGBX,NGBY)
          QP3G(NGBX,NGBY) = QG*Y3(NGBX,NGBY)
          QPRO4(NGBX,NGBY) = QG*Y4(NGBX,NGBY)
          QPRO5(NGBX,NGBY) = QG*Y5(NGBX,NGBY)
        ENDIF
      ENDIF
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
C
      SUBROUTINE HEATLOSS(I,J)
      INCLUDE 'DECLARATION'
      INTEGER I,J
C
      TDIF = THCDCAP/CRK
      DHL = DSQRT(TDIF*(TIME+DT))/2.0
      PTIHL = (PTTEMP(I,J) - TINIT(I,J))*PTDHL
      PTIHL = PTIHL + PTPHL(I,J)*(PTDHL**2)
      PTIHL = PTIHL + 2.0*PTQHL(I,J)*(PTDHL**3)
      T1 = 3.0*(DHL**2) + TDIF*DT
      T2 = TDIF*DT*(TEMP(I,J)-TINIT(I,J))/DHL + PTIHL
      PHL(I,J) = (T2 - (DHL**3)*(TEMP(I,J)-PTTEMP(I,J))/(TDIF*DT))/T1
      T1 = - TEMP(I,J) + TINIT(I,J)
      T1 = T1 + (DHL**2)*(TEMP(I,J)-PTTEMP(I,J))/(TDIF*DT)
      QHL(I,J) = (T1 + 2.0*PHL(I,J)*DHL)/(2.0*(DHL**2))
      T1 = THCDCAP*DHL/TDIF
      T2 = TEMP(I,J) - TINIT(I,J) + PHL(I,J)*DHL
      ENCAP(I,J) = T1*(T2 + 2.0*QHL(I,J)*(DHL**2))
      HLOSS(I,J) = ENCAP(I,J) - PTENCAP(I,J)
      HLOSS(I,J) = HLOSS(I,J)/DZ
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE EQUATIONCOKE(I,J)
      INCLUDE 'DECLARATION'
      INTEGER I,J,K
C
      D1  = ARHC*DEXP(-EENC/(GASCST*TEMP(I,J)))
      A2  = D1*PORE*DENO(I,J)*SO(I,J)*X2(I,J)*S8*DT
      A3  = ARHD*DEXP(-EEND/(GASCST*TEMP(I,J)))*Y5(I,J)*PRESG(I,J)*DT
      A1  = PTCC(I,J)
      CCO = PTCC(I,J)
      DO 10 K=1,100
        FX = -CCO + A2*(1.0-(CCO/COKMAX)**5.0) - A3*CCO + A1
        FDX = - 1.0 - (5.0*A2)*(1.0/COKMAX**5.0)*CCO**4.0 - A3
        CC1 = CCO - (FX/FDX)
        IF(DABS(CC1 - CCO).LT.1.0D-3) GOTO 20
        CCO = CC1
10    CONTINUE
      WRITE(11,390)I,J
      WRITE(31,390)I,J
```

```
390   FORMAT('FAILED CONVERGENCE IN THE COKE EQ.',5X,
     + 'Grid Block  = (',I2,',',I2,')')
      IFLCOK = 1
      GOTO 687
20    CONTINUE
      IF(CC1.LT.0.0) THEN
        CC1 = 0.0
      ENDIF
      CC(I,J)=CC1
      IF(CC(I,J).GT.COKMAX) CC(I,J)=COKMAX
687   CONTINUE
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE UPDATE
      INCLUDE 'DECLARATION'
      INTEGER I,J
C
      DO 5 J=1,NGBY
        DO 6 I=1,NGBX
          PTPRESG(I,J) = PRESG(I,J)
          PTTEMP(I,J) = TEMP(I,J)
          PTSG(I,J) = SG(I,J)
          PTSO(I,J) = SO(I,J)
          PTSW(I,J) = SW(I,J)
          PTY1(I,J) = Y1(I,J)
          PTY2(I,J) = Y2(I,J)
          PTY3(I,J) = Y3(I,J)
          PTY4(I,J) = Y4(I,J)
          PTY5(I,J) = Y5(I,J)
          PTX2(I,J) = X2(I,J)
          PTX3(I,J) = X3(I,J)
          PTDENG(I,J) = DENG(I,J)
          PTDENO(I,J) = DENO(I,J)
          PTDENW(I,J) = DENW(I,J)
          PTCC(I,J) = CC(I,J)
          PTUG(I,J) = UG(I,J)
          PTUO(I,J) = UO(I,J)
          PTUW(I,J) = UW(I,J)
          PTUR(I,J) = UR(I,J)
          PTHC(I,J) = HC(I,J)
6       CONTINUE
```

```
5     CONTINUE
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE FUNCT(I,J)
      INCLUDE 'DECLARATION'
      INTEGER I,IP1,IM1,J,JP1,JM1
C
      IP1 = I+1
      IF(I .EQ. NGBX) IP1 = I
      IM1 = I-1
      IF(I .EQ. 1) IM1 = I
      JP1 = J+1
      IF(J .EQ. NGBY) JP1 = J
      JM1 = J-1
      IF(J .EQ. 1) JM1 = J
      CALL PHYPROPERTIES(I,J)
      PDX = 6.328*PERM/DX
      PDY = 6.328*PERM/DY
      PDT = PORE/DT
C
      TGP = DENG(I,J)*AKRG(I,J)/VISG(I,J)
      TOP = DENO(I,J)*AKRO(I,J)/VISO(I,J)
      TWP = DENW(I,J)*AKRW(I,J)/VISW(I,J)
      TGMX = DENG(IM1,J)*AKRG(IM1,J)/VISG(IM1,J)
      TOMX = DENO(IM1,J)*AKRO(IM1,J)/VISO(IM1,J)
      TWMX = DENW(IM1,J)*AKRW(IM1,J)/VISW(IM1,J)
      TGMY = DENG(I,JM1)*AKRG(I,JM1)/VISG(I,JM1)
      TOMY = DENO(I,JM1)*AKRO(I,JM1)/VISO(I,JM1)
      TWMY = DENW(I,JM1)*AKRW(I,JM1)/VISW(I,JM1)
C
      PGPX = (PRESG(IP1,J)-PRESG(I,J))/DX
      POPX = (PRESO(IP1,J)-PRESO(I,J))/DX
      PWPX = (PRESW(IP1,J)-PRESW(I,J))/DX
      PGMX = (PRESG(I,J)-PRESG(IM1,J))/DX
      POMX = (PRESO(I,J)-PRESO(IM1,J))/DX
      PWMX = (PRESW(I,J)-PRESW(IM1,J))/DX
      PGPY = (PRESG(I,JP1)-PRESG(I,J))/DY
      POPY = (PRESO(I,JP1)-PRESO(I,J))/DY
      PWPY = (PRESW(I,JP1)-PRESW(I,J))/DY
      PGMY = (PRESG(I,J)-PRESG(I,JM1))/DY
      POMY = (PRESO(I,J)-PRESO(I,JM1))/DY
```

```
      PWMY = (PRESW(I,J)-PRESW(I,JM1))/DY
C
      GDGPX = DENG(I,J)*G*DDDX*WTMLG(I,J)
      ODGPX = DENO(I,J)*G*DDDX*WTMLO(I,J)
      WDGPX = DENW(I,J)*G*DDDX*WTMLW(I,J)
      GDGMX = DENG(IM1,J)*G*DDDX*WTMLG(IM1,J)
      ODGMX = DENO(IM1,J)*G*DDDX*WTMLO(IM1,J)
      WDGMX = DENW(IM1,J)*G*DDDX*WTMLW(IM1,J)
      GDGPY = DENG(I,J)*G*DDDY*WTMLG(I,J)
      ODGPY = DENO(I,J)*G*DDDY*WTMLO(I,J)
      WDGPY = DENW(I,J)*G*DDDY*WTMLW(I,J)
      GDGMY = DENG(I,JM1)*G*DDDY*WTMLG(I,JM1)
      ODGMY = DENO(I,JM1)*G*DDDY*WTMLO(I,JM1)
      WDGMY = DENW(I,JM1)*G*DDDY*WTMLW(I,JM1)
C
      Q1IP = QINJ1(I,J) - QPRO1(I,J) - QP1G(I,J)
      Q2IP = QINJ2(I,J) - QPRO2(I,J) - QP2G(I,J)
      Q3IP = QINJ3(I,J) - QPRO3(I,J) - QP3G(I,J)
      Q4IP = QINJ4(I,J) - QPRO4(I,J)
      Q5IP = QINJ5(I,J) - QPRO5(I,J)
C
      RR1 = S3*RA(I,J)+S6*RB(I,J)+S12*RD(I,J)
      RR2 = RB(I,J)+RC(I,J)
      RR3 = RA(I,J)-S7*RC(I,J)
      RR4 = S2*RA(I,J)+S5*RB(I,J)+S8 *RC(I,J)+S11*RD(I,J)
      RR5 = S1*RA(I,J)+S4*RB(I,J)+S10*RD(I,J)
C
      DSG = DENG(I,J)*SG(I,J)
      DSO = DENO(I,J)*SO(I,J)
      DSW = DENW(I,J)*SW(I,J)
      PTDSG = PTDENG(I,J)*PTSG(I,J)
      PTDSO = PTDENO(I,J)*PTSO(I,J)
      PTDSW = PTDENW(I,J)*PTSW(I,J)
C
C**** Water mass conservation equation
C
      WA1X = TGP*Y1(I,J)*PGPX  - TGMX*Y1(IM1,J)*PGMX
      WA2X = TGP*Y1(I,J)*GDGPX - TGMX*Y1(IM1,J)*GDGMX
      WA3X = TWP*PWPX  - TWMX*PWMX
      WA4X = TWP*WDGPX - TWMX*WDGMX
      WA1Y = TGP*Y1(I,J)*PGPY  - TGMY*Y1(I,JM1)*PGMY
      WA2Y = TGP*Y1(I,J)*GDGPY - TGMY*Y1(I,JM1)*GDGMY
      WA3Y = TWP*PWPY  - TWMY*PWMY
      WA4Y = TWP*WDGPY - TWMY*WDGMY
      WA5 = DSG*Y1(I,J)+DSW
```

```
      WA6 = PTDSG*PTY1(I,J)+PTDSW
      EQWX = PDX*(WA1X-WA2X + WA3X-WA4X)
      EQWY = PDY*(WA1Y-WA2Y + WA3Y-WA4Y)
      EQW = EQWX + EQWY + Q1IP + RR1 - PDT*(WA5-WA6)
C
C**** Heavy oil mass conservation equation
C
      OH1X = TGP*Y2(I,J)*PGPX  - TGMX*Y2(IM1,J)*PGMX
      OH2X = TGP*Y2(I,J)*GDGPX - TGMX*Y2(IM1,J)*GDGMX
      OH3X = TOP*X2(I,J)*POPX  - TOMX*X2(IM1,J)*POMX
      OH4X = TOP*X2(I,J)*ODGPX - TOMX*X2(IM1,J)*ODGMX
      OH1Y = TGP*Y2(I,J)*PGPY  - TGMY*Y2(I,JM1)*PGMY
      OH2Y = TGP*Y2(I,J)*GDGPY - TGMY*Y2(I,JM1)*GDGMY
      OH3Y = TOP*X2(I,J)*POPY  - TOMY*X2(I,JM1)*POMY
      OH4Y = TOP*X2(I,J)*ODGPY - TOMY*X2(I,JM1)*ODGMY
      OH5 = DSG*Y2(I,J)+DSO*X2(I,J)
      OH6 = PTDSG*PTY2(I,J)+PTDSO*PTX2(I,J)
      EQHOX = PDX*(OH1X-OH2X + OH3X-OH4X)
      EQHOY = PDY*(OH1Y-OH2Y + OH3Y-OH4Y)
      EQHO = EQHOX + EQHOY + Q2IP - RR2 - PDT*(OH5-OH6)
C
C**** Light oil mass conservation equation
C
      OL1X = TGP*Y3(I,J)*PGPX  - TGMX*Y3(IM1,J)*PGMX
      OL2X = TGP*Y3(I,J)*GDGPX - TGMX*Y3(IM1,J)*GDGMX
      OL3X = TOP*X3(I,J)*POPX  - TOMX*X3(IM1,J)*POMX
      OL4X = TOP*X3(I,J)*ODGPX - TOMX*X3(IM1,J)*ODGMX
      OL1Y = TGP*Y3(I,J)*PGPY  - TGMY*Y3(I,JM1)*PGMY
      OL2Y = TGP*Y3(I,J)*GDGPY - TGMY*Y3(I,JM1)*GDGMY
      OL3Y = TOP*X3(I,J)*POPY  - TOMY*X3(I,JM1)*POMY
      OL4Y = TOP*X3(I,J)*ODGPY - TOMY*X3(I,JM1)*ODGMY
      OL5 = DSG*Y3(I,J)+DSO*X3(I,J)
      OL6 = PTDSG*PTY3(I,J)+PTDSO*PTX3(I,J)
      EQLOX = PDX*(OL1X-OL2X + OL3X-OL4X)
      EQLOY = PDY*(OL1Y-OL2Y + OL3Y-OL4Y)
      EQLO = EQLOX + EQLOY + Q3IP - RR3 - PDT*(OL5-OL6)
C
C**** Inert Gas mass conservation equation
C
      GI1X = TGP*Y4(I,J)*PGPX  - TGMX*Y4(IM1,J)*PGMX
      GI2X = TGP*Y4(I,J)*GDGPX - TGMX*Y4(IM1,J)*GDGMX
      GI1Y = TGP*Y4(I,J)*PGPY  - TGMY*Y4(I,JM1)*PGMY
      GI2Y = TGP*Y4(I,J)*GDGPY - TGMY*Y4(I,JM1)*GDGMY
      GI3 = DSG*Y4(I,J)
      GI4 = PTDSG*PTY4(I,J)
```

```
      EQIGX = PDX*(GI1X-GI2X)
      EQIGY = PDY*(GI1Y-GI2Y)
      EQIG = EQIGX + EQIGY + Q4IP + RR4 - PDT*(GI3-GI4)
C
C**** Oxygen mass conservation equation
C
      OX1X = TGP*Y5(I,J)*PGPX  - TGMX*Y5(IM1,J)*PGMX
      OX2X = TGP*Y5(I,J)*GDGPX - TGMX*Y5(IM1,J)*GDGMX
      OX1Y = TGP*Y5(I,J)*PGPY  - TGMY*Y5(I,JM1)*PGMY
      OX2Y = TGP*Y5(I,J)*GDGPY - TGMY*Y5(I,JM1)*GDGMY
      OX3 = DSG*Y5(I,J)
      OX4 = PTDSG*PTY5(I,J)
      EQO2X = PDX*(OX1X-OX2X)
      EQO2Y = PDY*(OX1Y-OX2Y)
      EQO2 = EQO2X + EQO2Y + Q5IP - RR5 - PDT*(OX3-OX4)
C
C**** Energy conservation equation
C
      E1X = TGP*HG(I,J)*PGPX  - TGMX*HG(IM1,J)*PGMX
      E2X = TGP*HG(I,J)*GDGPX - TGMX*HG(IM1,J)*GDGMX
      E3X = TOP*HO(I,J)*POPX  - TOMX*HO(IM1,J)*POMX
      E4X = TOP*HO(I,J)*ODGPX - TOMX*HO(IM1,J)*ODGMX
      E5X = TWP*HW(I,J)*PWPX  - TWMX*HW(IM1,J)*PWMX
      E6X = TWP*HW(I,J)*WDGPX - TWMX*HW(IM1,J)*WDGMX
      E1Y = TGP*HG(I,J)*PGPY  - TGMY*HG(I,JM1)*PGMY
      E2Y = TGP*HG(I,J)*GDGPY - TGMY*HG(I,JM1)*GDGMY
      E3Y = TOP*HO(I,J)*POPY  - TOMY*HO(I,JM1)*POMY
      E4Y = TOP*HO(I,J)*ODGPY - TOMY*HO(I,JM1)*ODGMY
      E5Y = TWP*HW(I,J)*PWPY  - TWMY*HW(I,JM1)*PWMY
      E6Y = TWP*HW(I,J)*WDGPY - TWMY*HW(I,JM1)*WDGMY
      E7  = (QINJ1(I,J)*CWA + QINJ2(I,J)*CHO + QINJ3(I,J)*CLO +
     *QINJ4(I,J)*CIG + QINJ5(I,J)*CO2 )*(TINJ - TREF)
      E8 =(QP1G(I,J)+QP2G(I,J)+QP3G(I,J)+QPRO4(I,J)+QPRO5(I,J))*HG(I,J)
      E9 =(QPRO2(I,J) + QPRO3(I,J))*HO(I,J)
      E10 = QPRO1(I,J)*HW(I,J)
      E13X = (THCD/(DX*DX))*(TEMP(IP1,J)-2.0*TEMP(I,J)+TEMP(IM1,J))
      E13Y = (THCD/(DY*DY))*(TEMP(I,JP1)-2.0*TEMP(I,J)+TEMP(I,JM1))
      BH = BTHCD*(BHTEMP(I,J) - TEMP(I,J))
      BHEAT = DMAX1(BH,0.000)
      IF(IHLS .EQ. 1) THEN
        HLS = HLOSS(I,J)/DT
      ELSE
        HLS = 0.000
      ENDIF
      RH = RA(I,J)*HRA + RB(I,J)*HRB + RC(I,J)*HRC + RD(I,J)*HRD
```

```
      D1 = 1.0/DT
      E14= D1*(CC(I,J)*HC(I,J) - PTCC(I,J)*PTHC(I,J))
      D2 = (1.0-PORE)/DT
      E15= D2*(UR(I,J) - PTUR(I,J))
      E16= DSG*UG(I,J) + DSO*UO(I,J) + DSW*UW(I,J)
      E17= PTDSG*PTUG(I,J) + PTDSO*PTUO(I,J) + PTDSW*PTUW(I,J)
      E18= PDT*(E16 - E17)
      EQEBX = PDX*(E1X-E2X + E3X-E4X + E5X-E6X) + E13X
      EQEBY = PDY*(E1Y-E2Y + E3Y-E4Y + E5Y-E6Y) + E13Y
      EQEB = EQEBX + EQEBY +(E7-E8-E9-E10)+ BHEAT-HLS+RH-E14-E15-E18
C
      EALPR = EQIG + EQO2
      EALW = EQW
      EALO = EQHO + EQLO
      EALEN = EQEB
      EALY5 = EQO2
      EALX2 = EQHO
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE JACOBIAN
      INCLUDE 'DECLARATION'
      INTEGER I,J,K,IJ,M,L,LL
C
      MELIMF = 1
      DO 10 J=1,NGBY
        DO 9 I=1,NGBX
      CALL FUNCT(I,J)
      LL=NGBX*(J-1)+I
      L=(LL-1)*6+1
      B(L)   = -EALPR
      B(L+1) = -EALW
      B(L+2) = -EALO
      B(L+3) = -EALEN
      B(L+4) = -EALY5
      B(L+5) = -EALX2
C
      IF(ITNEWT .GE. 2) GOTO 4
      MELIMF = 0
      IJ=NGBX*(J-1)+I
      K=6*(IJ-1)+1
      IF(J.EQ.1) GOTO 1
```

```
      M=6*(IJ-NGBX-1)+1
C
      DELPR = PRESG(I,J-1)*0.02
      PRESG(I,J-1) = PRESG(I,J-1) + DELPR
      CALL FUNCT(I,J)
      A(K  ,M) = (B(L)  + EALPR)/DELPR
      A(K+1,M) = (B(L+1) + EALW)/DELPR
      A(K+2,M) = (B(L+2) + EALO)/DELPR
      A(K+3,M) = (B(L+3) + EALEN)/DELPR
      A(K+4,M) = (B(L+4) + EALY5)/DELPR
      A(K+5,M) = (B(L+5) + EALX2)/DELPR
      PRESG(I,J-1) = PRESG(I,J-1) - DELPR
      DELPR = 0.0
C
      IF(PTSW(I,J-1) .LT. 0.05)THEN
        DELSW = PTSW(I,J-1)*0.5
      ELSE
        DELSW = 0.01
      ENDIF
      SW(I,J-1) = SW(I,J-1) + DELSW
      CALL FUNCT(I,J)
      A(K  ,M+1) = (B(L) + EALPR)/DELSW
      A(K+1,M+1) = (B(L+1) + EALW)/DELSW
      A(K+2,M+1) = (B(L+2) + EALO)/DELSW
      A(K+3,M+1) = (B(L+3) + EALEN)/DELSW
      A(K+4,M+1) = (B(L+4) + EALY5)/DELSW
      A(K+5,M+1) = (B(L+5) + EALX2)/DELSW
      SW(I,J-1) = SW(I,J-1) - DELSW
      DELSW = 0.0
C
      IF(PTSO(I,J-1) .LT. 0.008)THEN
        DELSO = PTSO(I,J-1)*0.5
      ELSE
        DELSO = 0.01
      ENDIF
      SO(I,J-1) = SO(I,J-1) + DELSO
      CALL FUNCT(I,J)
      A(K  ,M+2) = (B(L) + EALPR)/DELSO
      A(K+1,M+2) = (B(L+1) + EALW)/DELSO
      A(K+2,M+2) = (B(L+2) + EALO)/DELSO
      A(K+3,M+2) = (B(L+3) + EALEN)/DELSO
      A(K+4,M+2) = (B(L+4) + EALY5)/DELSO
      A(K+5,M+2) = (B(L+5) + EALX2)/DELSO
      SO(I,J-1) = SO(I,J-1) - DELSO
      DELSO = 0.0
```

```
C
      DELTE = TEMP(I,J-1)*0.05
      TEMP(I,J-1) = TEMP(I,J-1) + DELTE
      CALL FUNCT(I,J)
      A(K  ,M+3) = (B(L) + EALPR)/DELTE
      A(K+1,M+3) = (B(L+1) + EALW)/DELTE
      A(K+2,M+3) = (B(L+2) + EALO)/DELTE
      A(K+3,M+3) = (B(L+3) + EALEN)/DELTE
      A(K+4,M+3) = (B(L+4) + EALY5)/DELTE
      A(K+5,M+3) = (B(L+5) + EALX2)/DELTE
      TEMP(I,J-1) = TEMP(I,J-1) - DELTE
      DELTE = 0.0
C
      IF(PTY5(I,J-1) .GT. 0.98) DELY5 = -0.0001
      IF(PTY5(I,J-1) .LE. 0.98) DELY5 = -0.01
      IF(PTY5(I,J-1) .LT. 0.01) DELY5 = PTY5(I,J-1)*0.5
      IF(PTY5(I,J-1) .LT. 0.0001) DELY5 = 0.0001
      Y5(I,J-1) = Y5(I,J-1) + DELY5
      CALL FUNCT(I,J)
      A(K  ,M+4) = (B(L) + EALPR)/DELY5
      A(K+1,M+4) = (B(L+1) + EALW)/DELY5
      A(K+2,M+4) = (B(L+2) + EALO)/DELY5
      A(K+3,M+4) = (B(L+3) + EALEN)/DELY5
      A(K+4,M+4) = (B(L+4) + EALY5)/DELY5
      A(K+5,M+4) = (B(L+5) + EALX2)/DELY5
      Y5(I,J-1) = Y5(I,J-1) - DELY5
      DELY5 = 0.0
C
      IF(PTX2(I,J-1) .LT. 0.008)THEN
        DELX2 = -PTX2(I,J-1)*0.5
      ELSE
        DELX2 = 0.0001
      ENDIF
      IF(PTX2(I,J-1) .GT. 0.990) DELX2 = -0.0001
      X2(I,J-1) = X2(I,J-1) + DELX2
      CALL FUNCT(I,J)
      A(K  ,M+5) = (B(L)   + EALPR)/DELX2
      A(K+1,M+5) = (B(L+1) + EALW)/DELX2
      A(K+2,M+5) = (B(L+2) + EALO)/DELX2
      A(K+3,M+5) = (B(L+3) + EALEN)/DELX2
      A(K+4,M+5) = (B(L+4) + EALY5)/DELX2
      A(K+5,M+5) = (B(L+5) + EALX2)/DELX2
      X2(I,J-1) = X2(I,J-1) - DELX2
      DELX2 = 0.0
C
```

```
      M=0
1     IF(I.EQ.1) GOTO 2
C
      DELPR = PRESG(I-1,J)*0.02
      PRESG(I-1,J) = PRESG(I-1,J) + DELPR
      CALL FUNCT(I,J)
      A(K  ,K-6) = (B(L)   + EALPR)/DELPR
      A(K+1,K-6) = (B(L+1) + EALW)/DELPR
      A(K+2,K-6) = (B(L+2) + EALO)/DELPR
      A(K+3,K-6) = (B(L+3) + EALEN)/DELPR
      A(K+4,K-6) = (B(L+4) + EALY5)/DELPR
      A(K+5,K-6) = (B(L+5) + EALX2)/DELPR
      PRESG(I-1,J) = PRESG(I-1,J) - DELPR
      DELPR = 0.0
C
      IF(PTSW(I-1,J) .LT. 0.05)THEN
        DELSW = PTSW(I-1,J)*0.5
      ELSE
        DELSW = 0.01
      ENDIF
      SW(I-1,J) = SW(I-1,J) + DELSW
      CALL FUNCT(I,J)
      A(K  ,K-5) = (B(L)   + EALPR)/DELSW
      A(K+1,K-5) = (B(L+1) + EALW)/DELSW
      A(K+2,K-5) = (B(L+2) + EALO)/DELSW
      A(K+3,K-5) = (B(L+3) + EALEN)/DELSW
      A(K+4,K-5) = (B(L+4) + EALY5)/DELSW
      A(K+5,K-5) = (B(L+5) + EALX2)/DELSW
      SW(I-1,J) = SW(I-1,J) - DELSW
      DELSW = 0.0
C
      IF(PTSO(I-1,J) .LT. 0.008)THEN
        DELSO = PTSO(I-1,J)*0.5
      ELSE
        DELSO = 0.01
      ENDIF
      SO(I-1,J) = SO(I-1,J) + DELSO
      CALL FUNCT(I,J)
      A(K  ,K-4) = (B(L)   + EALPR)/DELSO
      A(K+1,K-4) = (B(L+1) + EALW)/DELSO
      A(K+2,K-4) = (B(L+2) + EALO)/DELSO
      A(K+3,K-4) = (B(L+3) + EALEN)/DELSO
      A(K+4,K-4) = (B(L+4) + EALY5)/DELSO
      A(K+5,K-4) = (B(L+5) + EALX2)/DELSO
      SO(I-1,J) = SO(I-1,J) - DELSO
```

```
      DELSO = 0.0
C
      DELTE = TEMP(I-1,J)*0.05
      TEMP(I-1,J) = TEMP(I-1,J) + DELTE
      CALL FUNCT(I,J)
      A(K  ,K-3) = (B(L)   + EALPR)/DELTE
      A(K+1,K-3) = (B(L+1) + EALW)/DELTE
      A(K+2,K-3) = (B(L+2) + EALO)/DELTE
      A(K+3,K-3) = (B(L+3) + EALEN)/DELTE
      A(K+4,K-3) = (B(L+4) + EALY5)/DELTE
      A(K+5,K-3) = (B(L+5) + EALX2)/DELTE
      TEMP(I-1,J) = TEMP(I-1,J) - DELTE
      DELTE = 0.0
C
      IF(PTY5(I-1,J) .GT. 0.98) DELY5 = -0.0001
      IF(PTY5(I-1,J) .LE. 0.98) DELY5 = -0.01
      IF(PTY5(I-1,J) .LT. 0.01) DELY5 = PTY5(I-1,J)*0.5
      IF(PTY5(I-1,J) .LT. 0.0001) DELY5 = 0.0001
      Y5(I-1,J) = Y5(I-1,J) + DELY5
      CALL FUNCT(I,J)
      A(K  ,K-2) = (B(L)   + EALPR)/DELY5
      A(K+1,K-2) = (B(L+1) + EALW)/DELY5
      A(K+2,K-2) = (B(L+2) + EALO)/DELY5
      A(K+3,K-2) = (B(L+3) + EALEN)/DELY5
      A(K+4,K-2) = (B(L+4) + EALY5)/DELY5
      A(K+5,K-2) = (B(L+5) + EALX2)/DELY5
      Y5(I-1,J) = Y5(I-1,J) - DELY5
      DELY5 = 0.0
C
      IF(PTX2(I-1,J) .LT. 0.008)THEN
        DELX2 = -PTX2(I-1,J)*0.5
      ELSE
        DELX2 = 0.0001
      ENDIF
      IF(PTX2(I-1,J) .GT. 0.990) DELX2 = -0.0001
      X2(I-1,J) = X2(I-1,J) + DELX2
      CALL FUNCT(I,J)
      A(K  ,K-1) = (B(L)   + EALPR)/DELX2
      A(K+1,K-1) = (B(L+1) + EALW)/DELX2
      A(K+2,K-1) = (B(L+2) + EALO)/DELX2
      A(K+3,K-1) = (B(L+3) + EALEN)/DELX2
      A(K+4,K-1) = (B(L+4) + EALY5)/DELX2
      A(K+5,K-1) = (B(L+5) + EALX2)/DELX2
      X2(I-1,J) = X2(I-1,J) - DELX2
      DELX2 = 0.0
```

```
C
2     CONTINUE
C
      DELPR = PRESG(I,J)*0.02
      PRESG(I,J) = PRESG(I,J) + DELPR
      CALL FUNCT(I,J)
      A(K  ,K) = (B(L)   + EALPR)/DELPR
      A(K+1,K) = (B(L+1) + EALW)/DELPR
      A(K+2,K) = (B(L+2) + EALO)/DELPR
      A(K+3,K) = (B(L+3) + EALEN)/DELPR
      A(K+4,K) = (B(L+4) + EALY5)/DELPR
      A(K+5,K) = (B(L+5) + EALX2)/DELPR
      PRESG(I,J) = PRESG(I,J) - DELPR
      DELPR = 0.0
C
      IF(PTSW(I,J) .LT. 0.05)THEN
        DELSW = PTSW(I,J)*0.5
      ELSE
        DELSW = 0.01
      ENDIF
      SW(I,J) = SW(I,J) + DELSW
      CALL FUNCT(I,J)
      A(K  ,K+1) = (B(L)   + EALPR)/DELSW
      A(K+1,K+1) = (B(L+1) + EALW)/DELSW
      A(K+2,K+1) = (B(L+2) + EALO)/DELSW
      A(K+3,K+1) = (B(L+3) + EALEN)/DELSW
      A(K+4,K+1) = (B(L+4) + EALY5)/DELSW
      A(K+5,K+1) = (B(L+5) + EALX2)/DELSW
      SW(I,J) = SW(I,J) - DELSW
      DELSW = 0.0
C
      IF(PTSO(I,J) .LT. 0.008)THEN
        DELSO = PTSO(I,J)*0.5
      ELSE
        DELSO = 0.01
      ENDIF
      SO(I,J) = SO(I,J) + DELSO
      CALL FUNCT(I,J)
      A(K  ,K+2) = (B(L)   + EALPR)/DELSO
      A(K+1,K+2) = (B(L+1) + EALW)/DELSO
      A(K+2,K+2) = (B(L+2) + EALO)/DELSO
      A(K+3,K+2) = (B(L+3) + EALEN)/DELSO
      A(K+4,K+2) = (B(L+4) + EALY5)/DELSO
      A(K+5,K+2) = (B(L+5) + EALX2)/DELSO
      SO(I,J) = SO(I,J) - DELSO
```

```
      DELSO = 0.0
C
      DELTE = TEMP(I,J)*0.05
      TEMP(I,J) = TEMP(I,J) + DELTE
      CALL FUNCT(I,J)
      A(K  ,K+3) = (B(L)   + EALPR)/DELTE
      A(K+1,K+3) = (B(L+1) + EALW)/DELTE
      A(K+2,K+3) = (B(L+2) + EALO)/DELTE
      A(K+3,K+3) = (B(L+3) + EALEN)/DELTE
      A(K+4,K+3) = (B(L+4) + EALY5)/DELTE
      A(K+5,K+3) = (B(L+5) + EALX2)/DELTE
      TEMP(I,J) = TEMP(I,J) - DELTE
      DELTE = 0.0
C
      IF(PTY5(I,J) .GT. 0.98) DELY5 = -0.0001
      IF(PTY5(I,J) .LE. 0.98) DELY5 = -0.01
      IF(PTY5(I,J) .LT. 0.01) DELY5 = PTY5(I,J)*0.5
      IF(PTY5(I,J) .LT. 0.0001) DELY5 = 0.0001
      Y5(I,J) = Y5(I,J) + DELY5
      CALL FUNCT(I,J)
      A(K  ,K+4) = (B(L)   + EALPR)/DELY5
      A(K+1,K+4) = (B(L+1) + EALW)/DELY5
      A(K+2,K+4) = (B(L+2) + EALO)/DELY5
      A(K+3,K+4) = (B(L+3) + EALEN)/DELY5
      A(K+4,K+4) = (B(L+4) + EALY5)/DELY5
      A(K+5,K+4) = (B(L+5) + EALX2)/DELY5
      Y5(I,J) = Y5(I,J) - DELY5
      DELY5 = 0.0
C
      IF(PTX2(I,J) .LT. 0.008)THEN
        DELX2 = -PTX2(I,J)*0.5
      ELSE
        DELX2 = 0.0001
      ENDIF
      IF(PTX2(I,J) .GT. 0.990) DELX2 = -0.0001
      X2(I,J) = X2(I,J) + DELX2
      CALL FUNCT(I,J)
      A(K  ,K+5) = (B(L)   + EALPR)/DELX2
      A(K+1,K+5) = (B(L+1) + EALW)/DELX2
      A(K+2,K+5) = (B(L+2) + EALO)/DELX2
      A(K+3,K+5) = (B(L+3) + EALEN)/DELX2
      A(K+4,K+5) = (B(L+4) + EALY5)/DELX2
      A(K+5,K+5) = (B(L+5) + EALX2)/DELX2
      X2(I,J) = X2(I,J) - DELX2
      DELX2 = 0.0
```

```
C
      IF(I.EQ.NGBX)GOTO 3
C
      DELPR = PRESG(I+1,J)*0.02
      PRESG(I+1,J) = PRESG(I+1,J) + DELPR
      CALL FUNCT(I,J)
      A(K  ,K+6) = (B(L)   + EALPR)/DELPR
      A(K+1,K+6) = (B(L+1) + EALW)/DELPR
      A(K+2,K+6) = (B(L+2) + EALO)/DELPR
      A(K+3,K+6) = (B(L+3) + EALEN)/DELPR
      A(K+4,K+6) = (B(L+4) + EALY5)/DELPR
      A(K+5,K+6) = (B(L+5) + EALX2)/DELPR
      PRESG(I+1,J) = PRESG(I+1,J) - DELPR
      DELPR = 0.0
C
      IF(PTSW(I+1,J) .LT. 0.05)THEN
        DELSW = PTSW(I+1,J)*0.5
      ELSE
        DELSW = 0.01
      ENDIF
      SW(I+1,J) = SW(I+1,J) + DELSW
      CALL FUNCT(I,J)
      A(K  ,K+7) = (B(L)   + EALPR)/DELSW
      A(K+1,K+7) = (B(L+1) + EALW)/DELSW
      A(K+2,K+7) = (B(L+2) + EALO)/DELSW
      A(K+3,K+7) = (B(L+3) + EALEN)/DELSW
      A(K+4,K+7) = (B(L+4) + EALY5)/DELSW
      A(K+5,K+7) = (B(L+5) + EALX2)/DELSW
      SW(I+1,J) = SW(I+1,J) - DELSW
      DELSW = 0.0
C
      IF(PTSO(I+1,J) .LT. 0.008)THEN
        DELSO = PTSO(I+1,J)*0.5
      ELSE
        DELSO = 0.01
      ENDIF
      SO(I+1,J) = SO(I+1,J) + DELSO
      CALL FUNCT(I,J)
      A(K  ,K+8) = (B(L)   + EALPR)/DELSO
      A(K+1,K+8) = (B(L+1) + EALW)/DELSO
      A(K+2,K+8) = (B(L+2) + EALO)/DELSO
      A(K+3,K+8) = (B(L+3) + EALEN)/DELSO
      A(K+4,K+8) = (B(L+4) + EALY5)/DELSO
      A(K+5,K+8) = (B(L+5) + EALX2)/DELSO
```

```
      SO(I+1,J) = SO(I+1,J) - DELSO
      DELSO = 0.0
C
      DELTE = TEMP(I+1,J)*0.05
      TEMP(I+1,J) = TEMP(I+1,J) + DELTE
      CALL FUNCT(I,J)
      A(K  ,K+9) = (B(L)   + EALPR)/DELTE
      A(K+1,K+9) = (B(L+1) + EALW)/DELTE
      A(K+2,K+9) = (B(L+2) + EALO)/DELTE
      A(K+3,K+9) = (B(L+3) + EALEN)/DELTE
      A(K+4,K+9) = (B(L+4) + EALY5)/DELTE
      A(K+5,K+9) = (B(L+5) + EALX2)/DELTE
      TEMP(I+1,J) = TEMP(I+1,J) - DELTE
      DELTE = 0.0
C
      IF(PTY5(I+1,J) .GT. 0.98) DELY5 = -0.0001
      IF(PTY5(I+1,J) .LE. 0.98) DELY5 = -0.01
      IF(PTY5(I+1,J) .LT. 0.01) DELY5 = PTY5(I+1,J)*0.5
      IF(PTY5(I+1,J) .LT. 0.0001) DELY5 = 0.0001
      Y5(I+1,J) = Y5(I+1,J) + DELY5
      CALL FUNCT(I,J)
      A(K  ,K+10) = (B(L)   + EALPR)/DELY5
      A(K+1,K+10) = (B(L+1) + EALW)/DELY5
      A(K+2,K+10) = (B(L+2) + EALO)/DELY5
      A(K+3,K+10) = (B(L+3) + EALEN)/DELY5
      A(K+4,K+10) = (B(L+4) + EALY5)/DELY5
      A(K+5,K+10) = (B(L+5) + EALX2)/DELY5
      Y5(I+1,J) = Y5(I+1,J) - DELY5
      DELY5 = 0.0
C
      IF(PTX2(I+1,J) .LT. 0.008)THEN
        DELX2 = -PTX2(I+1,J)*0.5
      ELSE
        DELX2 = 0.0001
      ENDIF
      IF(PTX2(I+1,J) .GT. 0.990) DELX2 = -0.0001
      X2(I+1,J) = X2(I+1,J) + DELX2
      CALL FUNCT(I,J)
      A(K  ,K+11) = (B(L)   + EALPR)/DELX2
      A(K+1,K+11) = (B(L+1) + EALW)/DELX2
      A(K+2,K+11) = (B(L+2) + EALO)/DELX2
      A(K+3,K+11) = (B(L+3) + EALEN)/DELX2
      A(K+4,K+11) = (B(L+4) + EALY5)/DELX2
      A(K+5,K+11) = (B(L+5) + EALX2)/DELX2
      X2(I+1,J) = X2(I+1,J) - DELX2
```

```
      DELX2 = 0.0
C
3     IF(J.EQ.NGBY) GOTO 4
      M=6*(IJ+NGBX-1)+1
C
      DELPR = PRESG(I,J+1)*0.02
      PRESG(I,J+1) = PRESG(I,J+1) + DELPR
      CALL FUNCT(I,J)
      A(K  ,M) = (B(L)   + EALPR)/DELPR
      A(K+1,M) = (B(L+1) + EALW)/DELPR
      A(K+2,M) = (B(L+2) + EALO)/DELPR
      A(K+3,M) = (B(L+3) + EALEN)/DELPR
      A(K+4,M) = (B(L+4) + EALY5)/DELPR
      A(K+5,M) = (B(L+5) + EALX2)/DELPR
      PRESG(I,J+1) = PRESG(I,J+1) - DELPR
      DELPR = 0.0
C
      IF(PTSW(I,J+1) .LT. 0.05)THEN
        DELSW = PTSW(I,J+1)*0.5
      ELSE
        DELSW = 0.01
      ENDIF
      SW(I,J+1) = SW(I,J+1) + DELSW
      CALL FUNCT(I,J)
      A(K  ,M+1) = (B(L)   + EALPR)/DELSW
      A(K+1,M+1) = (B(L+1) + EALW)/DELSW
      A(K+2,M+1) = (B(L+2) + EALO)/DELSW
      A(K+3,M+1) = (B(L+3) + EALEN)/DELSW
      A(K+4,M+1) = (B(L+4) + EALY5)/DELSW
      A(K+5,M+1) = (B(L+5) + EALX2)/DELSW
      SW(I,J+1) = SW(I,J+1) - DELSW
      DELSW = 0.0
C
      IF(PTSO(I,J+1) .LT. 0.008)THEN
        DELSO = PTSO(I,J+1)*0.5
      ELSE
        DELSO = 0.01
      ENDIF
      SO(I,J+1) = SO(I,J+1) + DELSO
      CALL FUNCT(I,J)
      A(K  ,M+2) = (B(L) + EALPR)/DELSO
      A(K+1,M+2) = (B(L+1) + EALW)/DELSO
      A(K+2,M+2) = (B(L+2) + EALO)/DELSO
      A(K+3,M+2) = (B(L+3) + EALEN)/DELSO
      A(K+4,M+2) = (B(L+4) + EALY5)/DELSO
```

```
      A(K+5,M+2) = (B(L+5) + EALX2)/DELSO
      SO(I,J+1) = SO(I,J+1) - DELSO
      DELSO = 0.0
C
      DELTE = TEMP(I,J+1)*0.05
      TEMP(I,J+1) = TEMP(I,J+1) + DELTE
      CALL FUNCT(I,J)
      A(K  ,M+3) = (B(L)   + EALPR)/DELTE
      A(K+1,M+3) = (B(L+1) + EALW)/DELTE
      A(K+2,M+3) = (B(L+2) + EALO)/DELTE
      A(K+3,M+3) = (B(L+3) + EALEN)/DELTE
      A(K+4,M+3) = (B(L+4) + EALY5)/DELTE
      A(K+5,M+3) = (B(L+5) + EALX2)/DELTE
      TEMP(I,J+1) = TEMP(I,J+1) - DELTE
      DELTE = 0.0
C
      IF(PTY5(I,J+1) .GT. 0.98) DELY5 = -0.0001
      IF(PTY5(I,J+1) .LE. 0.98) DELY5 = -0.01
      IF(PTY5(I,J+1) .LT. 0.01) DELY5 = PTY5(I,J+1)*0.5
      IF(PTY5(I,J+1) .LT. 0.0001) DELY5 = 0.0001
      Y5(I,J+1) = Y5(I,J+1) + DELY5
      CALL FUNCT(I,J)
      A(K  ,M+4) = (B(L)   + EALPR)/DELY5
      A(K+1,M+4) = (B(L+1) + EALW)/DELY5
      A(K+2,M+4) = (B(L+2) + EALO)/DELY5
      A(K+3,M+4) = (B(L+3) + EALEN)/DELY5
      A(K+4,M+4) = (B(L+4) + EALY5)/DELY5
      A(K+5,M+4) = (B(L+5) + EALX2)/DELY5
      Y5(I,J+1) = Y5(I,J+1) - DELY5
      DELY5 = 0.0
C
      IF(PTX2(I,J+1) .LT. 0.008)THEN
        DELX2 = -PTX2(I,J+1)*0.5
      ELSE
        DELX2 = 0.0001
      ENDIF
      IF(X2(I,J+1) .GT. 0.990) DELX2 = -0.0001
      X2(I,J+1) = X2(I,J+1) + DELX2
      CALL FUNCT(I,J)
      A(K  ,M+5) = (B(L)   + EALPR)/DELX2
      A(K+1,M+5) = (B(L+1) + EALW)/DELX2
      A(K+2,M+5) = (B(L+2) + EALO)/DELX2
      A(K+3,M+5) = (B(L+3) + EALEN)/DELX2
      A(K+4,M+5) = (B(L+4) + EALY5)/DELX2
      A(K+5,M+5) = (B(L+5) + EALX2)/DELX2
```

```
      X2(I,J+1) = X2(I,J+1) - DELX2
      DELX2 = 0.0
C
      M=0
      K=0
4     CONTINUE
C
      IF(ITNEWT .EQ. 1) DAMP=0.15*URMX
      IF(ITNEWT .EQ. 2) DAMP=0.2*URMX
      IF(ITNEWT .EQ. 3) DAMP=0.3*URMX
      IF(ITNEWT .EQ. 4) DAMP=0.4*URMX
      IF(ITNEWT .EQ. 5) DAMP=0.5*URMX
      IF(ITNEWT .EQ. 6) DAMP=0.6*URMX
      IF(ITNEWT .EQ. 7) DAMP=0.7*URMX
      IF(ITNEWT .EQ. 8) DAMP=0.8*URMX
      IF(ITNEWT .EQ. 9) DAMP=0.9*URMX
      IF(ITNEWT .EQ.10) DAMP=URMX
C
      B(L)   = B(L)*DAMP
      B(L+1) = B(L+1)*DAMP
      B(L+2) = B(L+2)*DAMP
      B(L+3) = B(L+3)*DAMP
      B(L+4) = B(L+4)*DAMP
      B(L+5) = B(L+5)*DAMP
C
9     CONTINUE
10    CONTINUE
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE SOLVEAXB
      INCLUDE 'DECLARATION'
      INTEGER K
C
      IF(ITMET .EQ. 1) THEN
        CALL DIRECT
        GO TO 876
      ENDIF
C
      DO 222 K=1,N
        X0(K)=0.0D+00
222   CONTINUE
```

```
C
      IF(ISCAL .EQ. 1) CALL SCALING
C
      IF(NGBY .EQ. 1) THEN
        NBLLIN = 1
      ELSE
        NBLLIN = NGBX
      ENDIF
C
      IF(ILUF .EQ. 1 .AND. ITNEWT .EQ. 1) CALL ILUFACT
C
      IF(ITMET .EQ. 2 .AND. ILUF .EQ. 0) CALL GMRES
      IF(ITMET .EQ. 2 .AND. ILUF .EQ. 1) CALL GMRESPRE
      IF(ITMET .EQ. 3 .AND. ILUF .EQ. 0) CALL ORTHOMIN
      IF(ITMET .EQ. 3 .AND. ILUF .EQ. 1) CALL ORTHOMINPRE
      IF(ITMET .EQ. 4 .AND. ILUF .EQ. 0) CALL BICGSTAB
      IF(ITMET .EQ. 4 .AND. ILUF .EQ. 1) CALL BICGSTABPRE
C
876   CONTINUE
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE SCALING
      INCLUDE 'DECLARATION'
      INTEGER I,J
C
      IF(ITNEWT .GT. 1) GO TO 666
      DO 10 I=1,N
        EMAX = 0.0
        DO 20 J=1,N
          EMAX = DMAX1(EMAX,DABS(A(I,J)))
20      CONTINUE
        EM(I) = EMAX
10    CONTINUE
      DO 30 I=1,N
        DO 40 J=1,N
          A(I,J) = A(I,J)/EM(I)
40      CONTINUE
30    CONTINUE
666   CONTINUE
      DO 50 I=1,N
        B(I) = B(I)/EM(I)
```

```
50    CONTINUE
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE ILUFACT
      INCLUDE 'DECLARATION'
      INTEGER I,J,K,MINIJ,IB,JB,IJB
C
      DO 10 I=1,N
        IB = 1 + (I-1)/6
        DO 15 J=1,N
          JB = 1 + (J-1)/6
          MINIJ = MIN(I,J)
          T1 = 0.0
          DO 20 K=1,(MINIJ-1)
          T1 = T1 + W(I,K)*U(K,J)
20      CONTINUE
        SIJ = A(I,J) - T1
        IJB = IABS(IB-JB)
        IF(IJB .LE. 1 .OR. IJB .EQ. NBLLIN) THEN
          IF(I .GE. J) W(I,J) = SIJ
          IF(I .LT. J) U(I,J) = SIJ
        ELSE
          W(I,I) = W(I,I)
        ENDIF
15      CONTINUE
        U(I,I) = 1.0
        DO 25 J=(I+1),N
          U(I,J) = U(I,J)/W(I,I)
25      CONTINUE
10    CONTINUE
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE SOLVEILU
      INCLUDE 'DECLARATION'
      INTEGER I,K
C
      BD(1) = RHST(1)/W(1,1)
```

```
      DO 100 I=2,N
        T110 = 0.0
        DO 110 K=1,(I-1)
          T110 = T110 + W(I,K)*BD(K)
110     CONTINUE
        BD(I) = (RHST(I) - T110)/W(I,I)
100   CONTINUE
      SOL(N) = BD(N)/U(N,N)
      DO 120 I=(N-1),1,-1
        T120 = 0.0
        DO 125 K=(I+1),N
          T120 = T120 + U(I,K)*SOL(K)
125     CONTINUE
        SOL(I) = (BD(I) - T120)/U(I,I)
120   CONTINUE
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE DIRECT
      INCLUDE 'DECLARATION'
      INTEGER I,J,K,L,IROW,KCOL,KROW,MCOUNT,NPVT(MM)
C
      MCOUNT = 0
      PVMIN = 0.000001
      PVMAX = 999.0
      IF(MELIMF .EQ. 1) GOTO 999
      DO 33 I = 1,N
        DO 39 J = 1,N
          W(I,J) = A(I,J)
39      CONTINUE
        EM(I) = B(I)
        IF(A(I,I) .EQ. 0.0) THEN
          WRITE(11,*)'DIAG ELEM OF MATRIX A IS ZERO AT  I =',I
          WRITE(11,*)'NO UNIQUE SOLUTION EXISTS'
          GOTO 41
        ENDIF
33    CONTINUE
      DO 1 J=1,N-1
        NPVT(J) = J
        DO 2 I=J+1,N
          IF(DABS(W(NPVT(J),J)) .LT. DABS(W(I,J))) NPVT(J) = I
2       CONTINUE
```

```
        IF(NPVT(J) .EQ. J) GOTO 25
        DO 3 I=J,N
          SAVE = W(J,I)
          W(J,I) = W(NPVT(J),I)
          W(NPVT(J),I) = SAVE
3       CONTINUE
        SAVE = EM(J)
        EM(J) = EM(NPVT(J))
        EM(NPVT(J)) = SAVE
        PVMIN = DMAX1(PVMIN,DABS(W(J,J)))
        PVMAX = DMIN1(PVMAX,DABS(W(J,J)))
25      DO 32 IROW=J+1,N
          MCOUNT = MCOUNT + 1
          R(MCOUNT) = W(IROW,J)/W(J,J)
          IF(R(MCOUNT) .EQ. 0.0) GOTO 32
            DO 30 KCOL = J,N
              W(IROW,KCOL) = W(IROW,KCOL) - W(J,KCOL)*R(MCOUNT)
30          CONTINUE
            EM(IROW) = EM(IROW) - EM(J)*R(MCOUNT)
32      CONTINUE
1     CONTINUE
      GOTO 13
999   CONTINUE
      DO 8 K=1,N
        EM(K) = B(K)
8     CONTINUE
      DO 6 I=1,N-1
        SAVE = EM(I)
        EM(I) = EM(NPVT(I))
        EM(NPVT(I)) = SAVE
        DO 7 IROW = I+1,N
          MCOUNT = MCOUNT + 1
          EM(IROW) = EM(IROW) - EM(I)*R(MCOUNT)
7       CONTINUE
6     CONTINUE
13    CONTINUE
      DO 4 K = 1,N
        KROW = N+1-K
        DEL(KROW) = EM(KROW)/W(KROW,KROW)
        IF(KROW .EQ. N) GOTO 4
        SUBT = 0.0
        DO 5 L = KROW+1,N
          SUBT = W(KROW,L)*DEL(L) + SUBT
5       CONTINUE
          DEL(KROW) = (EM(KROW) - SUBT)/W(KROW,KROW)
```

```
4     CONTINUE
C
      RETURN
41    STOP
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE GMRES
      INCLUDE 'DECLARATION'
      INTEGER K,I,M,J,L,ICOUNT,IA,IB,IC,IR
C
      M=MGMRES
      ICOUNT = 0
      SQR0 = 0.0
      DO 10 K=1,N
        T1 = 0.0
        DO 15 L=1,N
          T1 = T1 + A(K,L) * X0(L)
15      CONTINUE
        V(K,1) = B(K) - T1
        SQR0 = SQR0 + (V(K,1)**2.0)
10    CONTINUE
      RONORM = DSQRT(SQR0)
      DO 20 K=1,N
        V(K,1) = V(K,1)/RONORM
20    CONTINUE
      ICOUNT = 0
1     CONTINUE
      ICOUNT = ICOUNT + 1
      IR = 1
      DO 30 J=1,M
        IR = IR + (J-1)
        DO 35 I=1,J
          T3 = 0.0
          DO 40 L=1,N
            T2 = 0.0
            DO 45 K=1,N
              T2 = T2 + A(L,K) * V(K,J)
45          CONTINUE
            T3 = T2*V(L,I) + T3
40        CONTINUE
          IF(J .EQ. 1) HJ = T3
          IF(J .EQ. 2) THEN
            IF(I .EQ. 1) HJM = T3
```

```
          IF(I .EQ. 2) HJ  = T3
        ENDIF
        IF(J .GT. 2) THEN
          IF(I .LE. (J-2)) H(IR+(I-1)) = T3
          IF(I .EQ. (J-1)) HJM = T3
          IF(I .EQ. J)     HJ  = T3
        ENDIF
35    CONTINUE
      SQVH = 0.0
      DO 70 L=1,N
        T4 = 0.0
        DO 75 K=1,N
          T4 = T4 + A(L,K)*V(K,J)
75      CONTINUE
        T5 = 0.0
        DO 80 K=1,J
          IF(J .EQ. 1) T5 = T5 + HJ*V(L,K)
          IF(J .EQ. 2) THEN
            IF(K .EQ. 1) T5 = T5 + HJM*V(L,K)
            IF(K .EQ. 2) T5 = T5 + HJ*V(L,K)
          ENDIF
          IF(J .GT. 2) THEN
            IF(K .LE. (J-2)) T5 = T5 + H(IR+(K-1))*V(L,K)
            IF(K .EQ. (J-1)) T5 = T5 + HJM*V(L,K)
            IF(K .EQ. J)     T5 = T5 + HJ*V(L,K)
          ENDIF
80      CONTINUE
        V(L,J+1) = T4 - T5
        SQVH = SQVH + V(L,J+1)**2.0
70    CONTINUE
      HJP = DSQRT(SQVH)
      DO 39 K=1,N
        V(K,J+1) = V(K,J+1)/HJP
39    CONTINUE
      IF(J .EQ. 1) THEN
        THETA = DATAN(-HJP/HJ)
        C1 = DCOS(THETA)
        S1 = DSIN(THETA)
        H(1) = C1*HJ - S1*HJP
        GOTO 200
      ENDIF
      IF(J .EQ. 2) THEN
        H(2) = C1*HJM - S1*HJ
        H(3) = DSQRT((S1*HJM + C1*HJ)**2 + HJP**2)
        C2 = (S1*HJM + C1*HJ)/H(3)
```

```
          S2 = -HJP/H(3)
          GOTO 200
        ENDIF
        H(IR+(J-2)) = C1*HJM - S1*HJ
        H(IR+(J-1)) = DSQRT((S1*HJM + C1*HJ)**2 + HJP**2)
        C2 = (S1*HJM + C1*HJ)/H(IR+(J-1))
        S2 = -HJP/H(IR+(J-1))
200     CONTINUE
        IF(J .EQ. 1) THEN
          GE(1) = C1*RONORM
          GE(2) = S1*RONORM
          GOTO 300
        ENDIF
        GE(J+1) = S2*GE(J)
        GE(J)   = C2*GE(J)
        S1 = S2
        C1 = C2
300     CONTINUE
        IF(J .EQ. M) THEN
          IB = IR+(M-1)
          GE(M) = GE(M)/H(IB)
          IC = IB
          IA = IB
          DO 600 K=(M-1),1,-1
            SUM = 0.0
            IA = IA - (K+1)
            IB = IC - (M-K)
            DO 601 L=M,(K+1),-1
              SUM = SUM + H(IB)*GE(L)
              IB = IB - (L-1)
601         CONTINUE
            GE(K) = (GE(K) - SUM)/H(IA)
600       CONTINUE
        ENDIF
30    CONTINUE
      AXNORM = 0.0
      DO 42 K=1,N
        T7 = 0.0
        DO 41 L=1,M
          T7 = T7 + V(K,L) * GE(L)
41      CONTINUE
        X0(K) = X0(K) + T7
        AXNORM = DMAX1(AXNORM,DABS(T7))
42    CONTINUE
      SQRM = 0.0
```

```
      DO 60 K=1,N
        T8 = 0.0
        DO 61 L=1,N
          T8 = T8 + A(K,L) * X0(L)
61      CONTINUE
        V(K,1) = B(K) - T8
        SQRM = SQRM + V(K,1)**2.0
60    CONTINUE
      RMNORM = DSQRT(SQRM)
      IF(RMNORM .LE. TOL) THEN
        DO 810 K=1,N
          DEL(K) = X0(K)
810     CONTINUE
        GO TO 991
      ENDIF
      IF(ICOUNT .GE. MAXITE) THEN
        DO 837 K=1,N
          DEL(K) = X0(K)
837     CONTINUE
        GO TO 98
      ENDIF
      DO 910 K=1,N
        V(K,1) = V(K,1)/RMNORM
910   CONTINUE
      RONORM = RMNORM
      GO TO 1
98    CONTINUE
991   CONTINUE
      ICITE = ICITE + ICOUNT
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE GMRESPRE
      INCLUDE 'DECLARATION'
      INTEGER K,I,M,J,L,ICOUNT,IA,IB,IC,IR
C
      M=MGMRES
      ICOUNT = 0
      SQR0 = 0.0
      DO 10 K=1,N
        T1 = 0.0
        DO 15 L=1,N
```

```
          T1 = T1 + A(K,L) * X0(L)
15      CONTINUE
        V(K,1) = B(K) - T1
        SQR0 = SQR0 + (V(K,1)**2.0)
10    CONTINUE
      RONORM = DSQRT(SQR0)
      DO 20 K=1,N
        V(K,1) = V(K,1)/RONORM
20    CONTINUE
      ICOUNT = 0
1     CONTINUE
      ICOUNT = ICOUNT + 1
      IR = 1
      DO 30 J=1,M
        IR = IR + (J-1)
        DO 311 K=1,N
          RHST(K) = V(K,J)
311     CONTINUE
        CALL SOLVEILU
        DO 31 L=1,N
          T2 = 0.0
          DO 32 K=1,N
            T2 = T2 + A(L,K) * SOL(K)
32        CONTINUE
          V(L,J+1) = T2
31      CONTINUE
        DO 35 I=1,J
          T3 = 0.0
          DO 40 L=1,N
            T3 = V(L,J+1)*V(L,I) + T3
40        CONTINUE
          IF(J .EQ. 1) HJ = T3
          IF(J .EQ. 2) THEN
            IF(I .EQ. 1) HJM = T3
            IF(I .EQ. 2) HJ  = T3
          ENDIF
          IF(J .GT. 2) THEN
            IF(I .LE. (J-2)) H(IR+(I-1)) = T3
            IF(I .EQ. (J-1)) HJM = T3
            IF(I .EQ. J)     HJ  = T3
          ENDIF
35      CONTINUE
        SQVH = 0.0
        DO 70 L=1,N
          T5 = 0.0
```

```
          DO 80 K=1,J
            IF(J .EQ. 1) T5 = T5 + HJ*V(L,K)
            IF(J .EQ. 2) THEN
              IF(K .EQ. 1) T5 = T5 + HJM*V(L,K)
              IF(K .EQ. 2) T5 = T5 + HJ*V(L,K)
            ENDIF
            IF(J .GT. 2) THEN
              IF(K .LE. (J-2)) T5 = T5 + H(IR+(K-1))*V(L,K)
              IF(K .EQ. (J-1)) T5 = T5 + HJM*V(L,K)
              IF(K .EQ. J)     T5 = T5 + HJ*V(L,K)
            ENDIF
80        CONTINUE
          V(L,J+1) = V(L,J+1) - T5
          SQVH = SQVH + V(L,J+1)**2.0
70      CONTINUE
        HJP = DSQRT(SQVH)
        DO 39 K=1,N
          V(K,J+1) = V(K,J+1)/HJP
39      CONTINUE
        IF(J .EQ. 1) THEN
          THETA = DATAN(-HJP/HJ)
          C1 = DCOS(THETA)
          S1 = DSIN(THETA)
          H(1) = C1*HJ - S1*HJP
          GOTO 200
        ENDIF
        IF(J .EQ. 2) THEN
          H(2) = C1*HJM - S1*HJ
          H(3) = DSQRT((S1*HJM + C1*HJ)**2 + HJP**2)
          C2 = (S1*HJM + C1*HJ)/H(3)
          S2 = -HJP/H(3)
          GOTO 200
        ENDIF
        H(IR+(J-2)) = C1*HJM - S1*HJ
        H(IR+(J-1)) = DSQRT((S1*HJM + C1*HJ)**2 + HJP**2)
        C2 = (S1*HJM + C1*HJ)/H(IR+(J-1))
        S2 = -HJP/H(IR+(J-1))
200     CONTINUE
        IF(J .EQ. 1) THEN
          GE(1) = C1*RONORM
          GE(2) = S1*RONORM
          GOTO 300
        ENDIF
        GE(J+1) = S2*GE(J)
        GE(J)   = C2*GE(J)
```

```
        S1 = S2
        C1 = C2
300     CONTINUE
        IF(J .EQ. M) THEN
          IB = IR+(M-1)
          GE(M) = GE(M)/H(IB)
          IC = IB
          IA = IB
          DO 600 K=(M-1),1,-1
            SUM = 0.0
            IA = IA - (K+1)
            IB = IC - (M-K)
            DO 601 L=M,(K+1),-1
              SUM = SUM + H(IB)*GE(L)
              IB = IB - (L-1)
601         CONTINUE
            GE(K) = (GE(K) - SUM)/H(IA)
600       CONTINUE
        ENDIF
30    CONTINUE
      DO 42 K=1,N
        T7 = 0.0
        DO 41 L=1,M
          T7 = T7 + V(K,L) * GE(L)
41      CONTINUE
        RHST(K) = T7
42    CONTINUE
      CALL SOLVEILU
      AXNORM = 0.0
      DO 402 K=1,N
        XO(K) = XO(K) + SOL(K)
        AXNORM = DMAX1(AXNORM,DABS(SOL(K)))
402   CONTINUE
      SQRM = 0.0
      DO 60 K=1,N
        T8 = 0.0
        DO 61 L=1,N
          T8 = T8 + A(K,L) * XO(L)
61      CONTINUE
        V(K,1) = B(K) - T8
        SQRM = SQRM + V(K,1)**2.0
60    CONTINUE
      RMNORM = DSQRT(SQRM)
      IF(RMNORM .LE. TOL) THEN
        DO 810 K=1,N
```

```
          DEL(K) = XO(K)
810     CONTINUE
        GO TO 991
      ENDIF
      IF(ICOUNT .GE. MAXITE) GO TO 98
      DO 910 K=1,N
        V(K,1) = V(K,1)/RMNORM
910   CONTINUE
      RONORM = RMNORM
      GO TO 1
98    CONTINUE
991   CONTINUE
      ICITE = ICITE + ICOUNT
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE BICGSTAB
      INCLUDE 'DECLARATION'
      INTEGER K,I,L
C
      DO 10 K=1,N
        T1 = 0.0
        DO 15 L=1,N
          T1 = T1 + A(K,L) * XO(L)
15      CONTINUE
        RO(K)  = B(K) - T1
        Q(K) = RO(K)
10    CONTINUE
      DO = 1.0
      AL = 1.0
      WO = 1.0
      DO 20 K=1,N
        VO(K) = 0.0
        PO(K) = 0.0
20    CONTINUE
      DO 30 I=1,MAXITE
        T5 = 0.0
        DO 35 K=1,N
          T5 = T5 + Q(K)*RO(K)
35      CONTINUE
        D1 = T5
        BET = (D1/DO)*(AL/WO)
```

```
      DO 40 K=1,N
        P1(K) = R0(K) + BET*(P0(K) - W0*V0(K))
40    CONTINUE
      DO 45 K=1,N
        T15 = 0.0
        DO 50 L=1,N
          T15 = T15 + A(K,L) * P1(L)
50      CONTINUE
        V1(K) = T15
45    CONTINUE
      T20 = 0.0
      DO 55 K=1,N
        T20 = T20 + Q(K)*V1(K)
55    CONTINUE
      DUM1 = T20
      AL = D1/DUM1
      DO 60 K=1,N
        S(K) = R0(K) - AL*V1(K)
60    CONTINUE
      DO 65 K=1,N
        T25 = 0.0
        DO 70 L=1,N
          T25 = T25 + A(K,L) * S(L)
70      CONTINUE
        T(K) = T25
65    CONTINUE
      T30 = 0.0
      DO 75 K=1,N
        T30 = T30 + T(K)*S(K)
75    CONTINUE
      DUM5 = T30
      T35 = 0.0
      DO 80 K=1,N
        T35 = T35 + T(K)*T(K)
80    CONTINUE
      DUM10 = T35
      W1 = DUM5/DUM10
      AXNORM = 0.0
      DO 85 K=1,N
        X1(K) = X0(K) + AL*P1(K) + W1*S(K)
        AXNORM = DMAX1(AXNORM,DABS(X1(K)-X0(K)))
85    CONTINUE
      SQRM1 = 0.0
      DO 119 K=1,N
        T119 = 0.0
```

```
          DO 125 L=1,N
            T119 = T119 + A(K,L) * X1(L)
125       CONTINUE
          RNO  = B(K) - T119
          SQRM1 = SQRM1 + RNO**2.0
119     CONTINUE
        RNORM = DSQRT(SQRM1)
        DO 90 K=1,N
          R1(K) = S(K) - W1*T(K)
90      CONTINUE
        SQRM = 0.0
        DO 95 K=1,N
          SQRM = SQRM + R1(K)**2.0
95      CONTINUE
        RMNORM = DSQRT(SQRM)
        IF(RNORM .LE. TOL)GO TO 991
        DO = D1
        WO = W1
        DO 180 K=1,N
          RO(K) = R1(K)
          PO(K) = P1(K)
          VO(K) = V1(K)
          XO(K) = X1(K)
180     CONTINUE
30    CONTINUE
      I = I - 1
98    CONTINUE
991   CONTINUE
      DO 810 K=1,N
        DEL(K) = X1(K)
810   CONTINUE
      ICITE = ICITE + I
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE BICGSTABPRE
      INCLUDE 'DECLARATION'
      INTEGER K,I,L
C
      DO 10 K=1,N
        T1 = 0.0
        DO 15 L=1,N
```

```
          T1 = T1 + A(K,L) * XO(L)
15      CONTINUE
        RO(K)  = B(K) - T1
        Q(K) = RO(K)
10    CONTINUE
      DO = 1.0
      AL = 1.0
      WO = 1.0
      DO 20 K=1,N
        VO(K) = 0.0
        PO(K) = 0.0
20    CONTINUE
      DO 30 I=1,MAXITE
        T5 = 0.0
        DO 35 K=1,N
          T5 = T5 + Q(K)*RO(K)
35      CONTINUE
        D1 = T5
        BET = (D1/DO)*(AL/WO)
        DO 40 K=1,N
          P1(K) = RO(K) + BET*(PO(K) - WO*VO(K))
          RHST(K) = P1(K)
40      CONTINUE
        CALL SOLVEILU
        DO 441K=1,N
          GE(K) = SOL(K)
441     CONTINUE
        DO 45 K=1,N
          T15 = 0.0
          DO 50 L=1,N
            T15 = T15 + A(K,L) * GE(L)
50        CONTINUE
          V1(K) = T15
45      CONTINUE
        T20 = 0.0
        DO 55 K=1,N
          T20 = T20 + Q(K)*V1(K)
55      CONTINUE
        DUM1 = T20
        AL = D1/DUM1
        DO 60 K=1,N
          S(K) = RO(K) - AL*V1(K)
          RHST(K) = S(K)
60      CONTINUE
        CALL SOLVEILU
```

```
      DO 442K=1,N
        Z(K) = SOL(K)
442   CONTINUE
      DO 65 K=1,N
        T25 = 0.0
        DO 70 L=1,N
          T25 = T25 + A(K,L) * Z(L)
70      CONTINUE
        T(K) = T25
65    CONTINUE
      T30 = 0.0
      DO 75 K=1,N
        T30 = T30 + T(K)*S(K)
75    CONTINUE
      DUM5 = T30
      T35 = 0.0
      DO 80 K=1,N
        T35 = T35 + T(K)*T(K)
80    CONTINUE
      DUM10 = T35
      W1 = DUM5/DUM10
      AXNORM = 0.0
      DO 85 K=1,N
        X1(K) = X0(K) + AL*GE(K) + W1*Z(K)
        AXNORM = DMAX1(AXNORM,DABS(X1(K)-X0(K)))
85    CONTINUE
      SQRM1 = 0.0
      DO 119 K=1,N
        T119 = 0.0
        DO 125 L=1,N
          T119 = T119 + A(K,L) * X1(L)
125     CONTINUE
        RNO  = B(K) - T119
        SQRM1 = SQRM1 + RNO**2.0
119   CONTINUE
      RNORM = DSQRT(SQRM1)
      DO 90 K=1,N
        R1(K) = S(K) - W1*T(K)
90    CONTINUE
      SQRM = 0.0
      DO 95 K=1,N
        SQRM = SQRM + R1(K)**2.0
95    CONTINUE
      RMNORM = DSQRT(SQRM)
      IF(RNORM .LE. TOL)GO TO 991
```

```
        D0 = D1
        W0 = W1
        DO 180 K=1,N
          R0(K) = R1(K)
          P0(K) = P1(K)
          V0(K) = V1(K)
          X0(K) = X1(K)
180     CONTINUE
30    CONTINUE
      I = I - 1
98    CONTINUE
991   CONTINUE
      DO 810 K=1,N
        DEL(K) = X1(K)
810   CONTINUE
      ICITE = ICITE + I
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE ORTHOMIN
      INCLUDE 'DECLARATION'
      INTEGER L,M,I,J,K,LN
C
      K = MGMRES
      DO 10 I=1,N
        T1 = 0.0
        DO 15 J=1,N
          T1 = T1 + A(I,J)*X0(J)
15      CONTINUE
        R0(I)  = B(I) - T1
        P(I,1) = R0(I)
10    CONTINUE
      DO 25 I=1,N
        T26 = 0.0
        DO 26 J=1,N
          T26 = T26 + A(I,J)*P(J,1)
26      CONTINUE
        AP(I,1) = T26
25    CONTINUE
      DO 19 M=1,MAXITE
        TOP = 0.0
        BOT = 0.0
```

```
      DO 40 I=1,N
        TOP = TOP + R0(I)*AP(I,M)
        BOT = BOT + AP(I,M)*AP(I,M)
40    CONTINUE
      ALFA = TOP/BOT
      DO 45 I=1,N
        X1(I) = X0(I) + ALFA*P(I,M)
        R1(I) = R0(I) - ALFA*AP(I,M)
45    CONTINUE
      DO 55 I=1,N
        T56 = 0.0
        DO 60 J=1,N
          T56 = T56 + A(I,J)*R1(J)
60      CONTINUE
        GE(I) = T56
55    CONTINUE
      LN = MAX0(1,(M-K+2))
      DO 50 L=LN,M
        TOP = 0.0
        BOT = 0.0
        DO 65 I=1,N
          TOP = TOP + GE(I)*AP(I,L)
          BOT = BOT + AP(I,L)*AP(I,L)
65      CONTINUE
        BETAN(L) = -TOP/BOT
50    CONTINUE
      DO 70 I=1,N
        SUM = 0.0
        DO 75 L=LN,N
          SUM = SUM + BETAN(L)*P(I,L)
75      CONTINUE
        P(I,M+1) = R1(I) + SUM
        SUM = 0.0
        DO 80 L=LN,N
          SUM = SUM + BETAN(L)*AP(I,L)
80      CONTINUE
        AP(I,M+1) = GE(I) + SUM
70     CONTINUE
      AXNORM = 0.0
      DO 95 I=1,N
        AXNORM = DMAX1(AXNORM,DABS(X1(I)-X0(I)))
95    CONTINUE
      SQRM1 = 0.0
      DO 119 I=1,N
        SQRM1 = SQRM1 + R1(I)**2.0
```

```
119     CONTINUE
        IF(RNORM .LE. TOL)GO TO 99
        DO 777 I=1,N
          X0(I) = X1(I)
          R0(I) = R1(I)
777     CONTINUE
19    CONTINUE
      M = M - 1
98    CONTINUE
99    CONTINUE
      DO 810 I=1,N
        DEL(I) = X1(I)
810   CONTINUE
      ICITE = ICITE + M
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE ORTHOMINPRE
      INCLUDE 'DECLARATION'
      INTEGER L,M,I,J,K,LN
C
      K = MGMRES
      DO 10 I=1,N
        T1 = 0.0
        DO 15 J=1,N
          T1 = T1 + A(I,J)*X0(J)
15      CONTINUE
        R0(I)  = B(I) - T1
        RHST(I) = R0(I)
10    CONTINUE
      CALL SOLVEILU
      DO 20 I=1,N
        R0(I) = SOL(I)
        P(I,1) = R0(I)
20    CONTINUE
      DO 25 I=1,N
        T26 = 0.0
        DO 26 J=1,N
          T26 = T26 + A(I,J)*P(J,1)
26      CONTINUE
        AP(I,1) = T26
25    CONTINUE
```

```
      DO 27 I=1,N
        RHST(I) = AP(I,1)
27    CONTINUE
      CALL SOLVEILU
      DO 28 I=1,N
         AP(I,1) =  SOL(I)
28    CONTINUE
      DO 19 M=1,MAXITE
        TOP = 0.0
        BOT = 0.0
        DO 40 I=1,N
          TOP = TOP + RO(I)*AP(I,M)
          BOT = BOT + AP(I,M)*AP(I,M)
40      CONTINUE
        ALFA = TOP/BOT
        DO 45 I=1,N
          X1(I) = XO(I) + ALFA*P(I,M)
          R1(I) = RO(I) - ALFA*AP(I,M)
45      CONTINUE
        DO 55 I=1,N
          T56 = 0.0
          DO 60 J=1,N
            T56 = T56 + A(I,J)*R1(J)
60        CONTINUE
          GE(I) = T56
          RHST(I) = GE(I)
55      CONTINUE
        CALL SOLVEILU
        DO 57 I=1,N
          GE(I) = SOL(I)
57      CONTINUE
        LN = MAXO(1,(M-K+2))
        DO 50 L=LN,M
          TOP = 0.0
          BOT = 0.0
          DO 65 I=1,N
            TOP = TOP + GE(I)*AP(I,L)
            BOT = BOT + AP(I,L)*AP(I,L)
65        CONTINUE
          BETAN(L) = -TOP/BOT
50      CONTINUE
        DO 70 I=1,N
          SUM = 0.0
          DO 75 L=LN,N
            SUM = SUM + BETAN(L)*P(I,L)
```

```
75          CONTINUE
            P(I,M+1) = R1(I) + SUM
            SUM = 0.0
            DO 80 L=LN,N
              SUM = SUM + BETAN(L)*AP(I,L)
80          CONTINUE
            AP(I,M+1) = GE(I) + SUM
70        CONTINUE
          AXNORM = 0.0
          DO 95 I=1,N
            AXNORM = DMAX1(AXNORM,DABS(X1(I)-X0(I)))
95        CONTINUE
          SQRM1 = 0.0
          DO 119 I=1,N
            SQRM1 = SQRM1 + R1(I)**2.0
119       CONTINUE
          RNORM = DSQRT(SQRM1)
          IF(RNORM .LE. TOL)GO TO 99
          DO 777 I=1,N
            X0(I) = X1(I)
            R0(I) = R1(I)
777       CONTINUE
19      CONTINUE
        M = M - 1
98      CONTINUE
99      CONTINUE
        DO 810 I=1,N
          DEL(I) = X1(I)
810     CONTINUE
        ICITE = ICITE + M
C
        RETURN
        END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
        SUBROUTINE CONVERGENCE
        INCLUDE 'DECLARATION'
        INTEGER I,J,IJ,M
C
        IFLCF1 = 0
        IFLCF2 = 0
        DO 2 J=1,NGBY
          DO 1 I=1,NGBX
            IJ=NGBX*(J-1)+I
```

```
      M = (IJ-1)*6
      IF(DABS(DEL(M+1)) .LT. 1.0D-18) DEL(M+1)=0.00000
      IF(DABS(DEL(M+2)) .LT. 1.0D-18) DEL(M+2)=0.00000
      IF(DABS(DEL(M+3)) .LT. 1.0D-18) DEL(M+3)=0.00000
      IF(DABS(DEL(M+4)) .LT. 1.0D-18) DEL(M+4)=0.00000
      IF(DABS(DEL(M+5)) .LT. 1.0D-18) DEL(M+5)=0.00000
      IF(DABS(DEL(M+6)) .LT. 1.0D-18) DEL(M+6)=0.00000
C
      PRESG(I,J) = PRESG(I,J) + DEL(M+1)
      SW(I,J)    = SW(I,J)    + DEL(M+2)
      SO(I,J)    = SO(I,J)    + DEL(M+3)
      TEMP(I,J)  = TEMP(I,J)  + DEL(M+4)
      Y5(I,J)    = Y5(I,J)    + DEL(M+5)
      X2(I,J)    = X2(I,J)    + DEL(M+6)
C
      IF(PRESG(I,J) .LT. 58.58) THEN
        IFLCF1 = 1
        GOTO 3
      ENDIF
      CTEST1 = DABS(DEL(M+1)/PRESG(I,J))
      IF(CTEST1 .GT. PCON) THEN
        IFLCF2 = 1
      ENDIF
C
      CTEST2 = DABS(DEL(M+2))
      IF(SW(I,J) .LT. 0.000) THEN
        IF(PTSW(I,J) .LT. 1.0D-04) THEN
          SW(I,J)=PTSW(I,J)*0.5
          CTEST2 = 0.00
        ELSE
          IFLCF1 = 1
          GOTO 3
        ENDIF
      ENDIF
      IF(CTEST2 .GT. SWCON) THEN
        IFLCF2 = 1
      ENDIF
C
      CTEST3 = DABS(DEL(M+3))
      IF(SO(I,J) .LT. 0.0000) THEN
        IF(PTSO(I,J) .LT. 1.0D-04) THEN
          SO(I,J)=PTSO(I,J)*0.5
          CTEST3 = 0.0
        ELSE
          IFLCF1 = 1
```

```
          GOTO 3
        ENDIF
      ENDIF
      IF(CTEST3 .GT. SOCON) THEN
        IFLCF2 = 1
      ENDIF
C
      IF(TEMP(I,J) .LT. 540.0) THEN
        IFLCF1 = 1
        GOTO 3
      ENDIF
      CTEST4 = DABS(DEL(M+4)/TEMP(I,J))
      IF(CTEST4 .GT. TCON) THEN
        IFLCF2 = 1
      ENDIF
C
      CTEST5 = DABS(DEL(M+5))
      IF(Y5(I,J) .LT. 0.0000) THEN
        IF(PTY5(I,J) .LT. 1.0D-04) THEN
          Y5(I,J)=PTY5(I,J)
          CTEST5 = 0.0
        ELSE
          IFLCF1 = 1
          GOTO 3
        ENDIF
      ENDIF
      IF(Y5(I,J) .GT. 1.0000) THEN
        IF(PTY5(I,J) .GT. 0.9990) THEN
          Y5(I,J) = DSQRT(PTY5(I,J))
          CTEST5 = 0.0
        ELSE
          IFLCF1 = 1
          GOTO 3
        ENDIF
      ENDIF
      IF(CTEST5 .GT. YCON) THEN
        IFLCF2 = 1
      ENDIF
C
      CTEST6 = DABS(DEL(M+6))
      IF(X2(I,J) .GT. 1.00) THEN
        IF(PTX2(I,J) .GT. 0.99) THEN
          X2(I,J) = DSQRT(PTX2(I,J))
          CTEST6 = 0.0
        ELSE
```

```
              IFLCF1 = 1
              GOTO 3
            ENDIF
          ENDIF
          IF(CTEST6 .GT. XCON) THEN
            IFLCF2 = 1
          ENDIF
1       CONTINUE
2     CONTINUE
3     CONTINUE
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE RESET
      INCLUDE 'DECLARATION'
      INTEGER I,J
C
      DO 5 I=1,NGBX
        DO 6 J=1,NGBY
          PRESG(I,J) = PTPRESG(I,J)
          SW(I,J) = PTSW(I,J)
          SO(I,J) = PTSO(I,J)
          TEMP(I,J) = PTTEMP(I,J)
          Y5(I,J) = PTY5(I,J)
          X2(I,J) = PTX2(I,J)
          CC(I,J) = PTCC(I,J)
6       CONTINUE
5     CONTINUE
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE TIMESTEP
      INCLUDE 'DECLARATION'
      INTEGER I,J
C
      DOMAX = 0.0
      DWMAX = 0.0
      DPMAX = 0.0
      DEMAX = 0.0
```

```
      DXMAX = 0.0
      DYMAX = 0.0
C
      DO 1 I=1,NGBX
        DO 2 J=1,NGBY
          DO = DABS(SO(I,J)-PTSO(I,J))
          DOMAX = DMAX1(DOMAX,DO)
          DW = DABS(SW(I,J)-PTSW(I,J))
          DWMAX = DMAX1(DWMAX,DW)
          DP = DABS(PRESG(I,J)-PTPRESG(I,J))
          DPMAX = DMAX1(DPMAX,DP)
          DTEMP = DABS(TEMP(I,J)-PTTEMP(I,J))
          DEMAX = DMAX1(DEMAX,DTEMP)
          DX2 = DABS(X2(I,J)-PTX2(I,J))
          DXMAX = DMAX1(DXMAX,DX2)
          DY5 = DABS(Y5(I,J)-PTY5(I,J))
          DYMAX = DMAX1(DYMAX,DY5)
2       CONTINUE
1     CONTINUE
C
      SNEW = 2.0*SONORM*DT/(SONORM+DOMAX)
      TNEW = 2.0*SWNORM*DT/(SWNORM+DWMAX)
      SNEW = DMIN1(SNEW,TNEW)
      TNEW = 2.0*PNORM*DT/(PNORM+DPMAX)
      SNEW = DMIN1(SNEW,TNEW)
      TNEW = 2.0*TNORM*DT/(TNORM+DEMAX)
      SNEW = DMIN1(SNEW,TNEW)
      TNEW = 2.0*XNORM*DT/(XNORM+DXMAX)
      SNEW = DMIN1(SNEW,TNEW)
      TNEW = 2.0*YNORM*DT/(YNORM+DYMAX)
      SNEW = DMIN1(SNEW,TNEW)
C
      DT = SNEW
      DUMDT = TIME + DT
      IF(TIME .LT. PRHT .AND. DUMDT .GT. PRHT) DT = PRHT - TIME
      IF(DUMDT .GT. ETIM) DT = ETIM - TIME
      IF(DT .GT. DTMAX) DT=DTMAX
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE OUTPUT
      INCLUDE 'DECLARATION'
```

```
C
      IF(ITSTEP .EQ. 0) CALL OUTPUT1
      CALL OUTPUT2
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE OUTPUT1
      INCLUDE 'DECLARATION'
C
      WRITE(11,601) IRUNNO
      WRITE(11,602)RLEN,RWID,DZ,NGBX,NGBY,(NGBX/NGBX)
      WRITE(11,325)DTINIT,DTMAX,ETIM,OUTFQD,IOUTFQ,MSTEP,MXIT,
     +             PNORM,PCON,SWNORM,SWCON,SONORM,SOCON,TNORM,TCON,
     +             YNORM,YCON,XNORM,XCON
      WRITE(11,327)
      IF(ITMET .EQ. 1) THEN
        WRITE(11,801)
        WRITE(11,820)
        GO TO 830
      ENDIF
      IF(ITMET .EQ. 2)THEN
        WRITE(11,802)MGMRES
      ENDIF
      IF(ITMET .EQ. 3)THEN
        WRITE(11,803)MGMRES
      ENDIF
      IF(ITMET .EQ. 4)THEN
        WRITE(11,804)
      ENDIF
      IF(ILUF .EQ. 1) THEN
        WRITE(11,805)
      ELSE
        WRITE(11,806)
      ENDIF
      IF(ISCAL .EQ. 1) THEN
        WRITE(11,807)
      ELSE
        WRITE(11,808)
      ENDIF
      WRITE(11,809)MAXITE,TOL
830   CONTINUE
      WRITE(11,605)
```

```
      WRITE(11,606)
C
601   FORMAT(//5X,63('*')/5X,'*',61X,'*'/6('*'),1X,
     +'OUTPUT FILE OF THE IN-SITU COMBUSTION SIMULATOR :"OUTPUT.N"',
     +1X,5('*')/5X,'*',61X,'*'/5X,63('*')/
     +//30X,12('*')/30X,'RUN NO. =',I3/30X,12('*')///)
602   FORMAT(27('*'),2X,'RESERVOIR DATA',2X,27('*')//
     +'Reservoir Length (x-direction)',8X,'=',F9.3,2X,'ft'/
     +'Reservoir Width  (y-direction)',8X,'=',F9.3,2X,'ft'/
     +'Reservoir Height (z-direction)',8X,'=',F9.3,2X,'ft'/
     +'Number of Grid Blocks in x-direction ',1X,'=',I9/
     +'Number of Grid Blocks in y-direction ',1X,'=',I9/
     +'Number of Grid Blocks in z-direction ',1X,'=',I9)
605     FORMAT('KEY'/72('-')/
     +'TSN  : Time Step No.'/
     +'TSS  : Time Step Size'/
     +'RRT  : Real Reservoir Time'/
     +'NIPTS: Newton Iteration Per Time Step'/
     +'LIPTS: Linear Iteration Per Time Step'/
     +'TLI  : Total Linear Iteration'/
     +'TNI  : Total Newton Iteration'/
     +'FFL1 : Fail Flag 1 (Max number of Newton Iteration exceeded)'/
     +'FFL2 : Fail Flag 2 (Reselts from the Newton Iteration are'
     +' unacceptable)'//72('*')///)
606   FORMAT('TSN',8X,'TSS',10X,'RRT',5X,'NIPTS  LIPTS',5X,'TLI ',
     +7X,'TNI',6X,'FFL1',3X,'FFL2 '/
     +4('-'),4X,10('-'),3X,8('-'),3X,5('-'),2X,5('-'),3X,7('-'),4X,
     +6('-'),4X,5('-'),2X,5('-'))
325   FORMAT(///23('*'),2X,'NUMERICAL CONTROL DATA',2X,23('*')//
     +'Initial Timestep',15X,'=',F8.3,2X,'days'/
     +'Maximum Timestep',15X,'=',F8.3,2X,'days'/
     +'Stopping End Time',14X,'=',F8.3,2X,'days'/
     +'Output Frequency (day)',9X,'=',F8.3,2X,'days'/
     +'Output Frequency (timestep)',4X,'=',I8/
     +'Maximum Number of Timestep',5X,'=',I8/
     +'Max. Newton Iter. Per Timestep',1X,'=',I8///
     +'Timestep Change Norms',24X,'Convergence Tolerances'/
     + 26('-'),19X,24('-')/
     +'Pressure',5X,'=',F7.2,2X,'psi',19X,'Pressure',5X,'=',F9.5/
     +'Water Satur.',1X,'=',F7.2,2X,22X,'Water Satur.',1X,'=',F9.5/
     +'Oil Satur.',3X,'=',F7.2,2X,22X,'Oil Satur.',3X,'=',F9.5/
     +'Temperature',2X,'=',F7.2,2X,'F',21X,'Temperature',2X,'=',F9.5/
     +'Oxygen',7X,'=',F7.2,2X,22X,'Oxygen',7X,'=',F9.5/
     +'Heavy Oil',4X,'=',F7.2,2X,22X,'Heavy Oil',4X,'=',F9.5)
327   FORMAT(//'Data for Solving Linear System'/30('-'))
```

```
801   FORMAT('Solution Method = LU Decomposition')
820   FORMAT('Preconditioning = --'/'Scaling         = --'/
     +'Max. No.of Iter.= --'/'Tolerance       = --'//)
802   FORMAT('Solution Method = GMRES(',I2,')')
803   FORMAT('Solution Method = ORTHOMIN(',I2,')')
804   FORMAT('Solution Method = BI-CGSTAB')
805   FORMAT('Preconditioning = Yes')
806   FORMAT('Preconditioning = No')
807   FORMAT('Scaling         = Yes')
808   FORMAT('Scaling         = No')
809   FORMAT('Max. No.of Iter.= ',I3/ 'Tolerance       = ',D7.1//)
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
      SUBROUTINE OUTPUT2
      INCLUDE 'DECLARATION'
      INTEGER I,J
C
      IF(ITSTEP .EQ. 0) THEN
        IF(IOUT3 .EQ. 1) THEN
          WRITE(31,310) IRUNNO
        ELSE
          WRITE(31,311) IRUNNO
        ENDIF
        WRITE(31,315)RLEN,RWID,DZ,NGBX,NGBY,(NGBX/NGBX),DX,DY,DZ,
     +  SLOPEX,SLOPEY,PERM,PORE
        WRITE(31,320)QINJ1(1,1)*VOL,QINJ2(1,1)*VOL,QINJ3(1,1)*VOL,
     +  QINJ4(1,1)*VOL,QINJ5(1,1)*VOL,TINJ-459.69
        WRITE(31,321)
        WRITE(31,322) PRHT,BTHCD
        WRITE(31,323) ((BHTEMP(I,J)-459.69,I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,325)DTINIT,DTMAX,ETIM,OUTFQD,IOUTFQ,MSTEP,MXIT,
     +  PNORM,PCON,SWNORM,SWCON,SONORM,SOCON,TNORM,TCON,
     +  YNORM,YCON,XNORM,XCON
        WRITE(31,327)
        IF(ITMET .EQ. 1) THEN
          WRITE(31,801)
          WRITE(31,820)
          GO TO 840
        ENDIF
        IF(ITMET .EQ. 2)THEN
          WRITE(31,802)MGMRES
```

```
        ENDIF
        IF(ITMET .EQ. 3)THEN
          WRITE(31,803)MGMRES
        ENDIF
        IF(ITMET .EQ. 4)THEN
          WRITE(31,804)
        ENDIF
        IF(ILUF .EQ. 1)THEN
          WRITE(31,805)
        ELSE
          WRITE(31,806)
        ENDIF
        IF(ISCAL .EQ. 1)THEN
          WRITE(31,807)
        ELSE
          WRITE(31,808)
        ENDIF
        WRITE(31,809)MAXITE,TOL
840     CONTINUE
        WRITE(31,330)
      ENDIF
      IF(ITSTEP .EQ. 0) THEN
        DTSTEP = 0.0
      ELSE
        DTSTEP = DT
      ENDIF
      PWW = QPRO1(NGBX,NGBY)*VOL
      POH = QPRO2(NGBX,NGBY)*VOL
      POL = QPRO3(NGBX,NGBY)*VOL
      PGW = QP1G(NGBX,NGBY)*VOL
      PGH = QP2G(NGBX,NGBY)*VOL
      PGL = QP3G(NGBX,NGBY)*VOL
      PGI = QPRO4(NGBX,NGBY)*VOL
      PGO = QPRO5(NGBX,NGBY)*VOL
      WRITE(31,335) TIME,DTSTEP,ITSTEP
      WRITE(31,500) ((PRESG(I,J),I=1,NGBX),J=NGBY,1,-1)
      IF(IOUT3 .EQ. 1) THEN
        WRITE(31,501) ((PRESO(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,502) ((PRESW(I,J),I=1,NGBX),J=NGBY,1,-1)
      ENDIF
      IF(IOUT3 .EQ. 1) THEN
        WRITE(31,503) ((SG(I,J),I=1,NGBX),J=NGBY,1,-1)
      ENDIF
      WRITE(31,504) ((SO(I,J),I=1,NGBX),J=NGBY,1,-1)
      WRITE(31,505) ((SW(I,J),I=1,NGBX),J=NGBY,1,-1)
```

```
      WRITE(31,506) ((TEMP(I,J)-459.69,I=1,NGBX),J=NGBY,1,-1)
      IF(IOUT3 .EQ. 1) THEN
        WRITE(31,507) ((Y1(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,508) ((Y2(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,509) ((Y3(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,510) ((Y4(I,J),I=1,NGBX),J=NGBY,1,-1)
      ENDIF
      WRITE(31,511) ((Y5(I,J),I=1,NGBX),J=NGBY,1,-1)
      WRITE(31,512) ((X2(I,J),I=1,NGBX),J=NGBy,1,-1)
      IF(IOUT3 .EQ. 1) THEN
        WRITE(31,513) ((X3(I,J),I=1,NGBX),J=NGBy,1,-1)
      ENDIF
      WRITE(31,514) ((CC(I,J),I=1,NGBX),J=NGBY,1,-1)
      IF(IOUT3 .EQ. 1) THEN
        WRITE(31,515) ((DENG(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,516) ((DENO(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,517) ((DENW(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,518) ((VISG(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,519) ((VISO(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,520) ((VISW(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,521) ((EK1(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,522) ((EK2(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,523) ((EK3(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,524) ((AKRG(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,525) ((AKRO(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,526) ((AKRW(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,527) ((RA(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,528) ((RB(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,529) ((RC(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,530) ((RD(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,531) ((UG(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,532) ((UO(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,533) ((UW(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,534) ((UR(I,J),I=1,NGBX),J=NGBY,1,-1)
        WRITE(31,535) ((HC(I,J),I=1,NGBX),J=NGBY,1,-1)
      ENDIF
      WRITE(31,375) PWW,PGW,PWW+PGW,POH,PGH,POH+PGH,POL,PGL,POL+PGL,
     +PGI,PGI,PGO,PGO
      WRITE(31,380)
C
310   FORMAT(//5X,63('*')/5X,'*',61X,'*'/6('*'),1X,
     +'OUTPUT FILE OF THE IN-SITU COMBUSTION SIMULATOR :"OUTPUT.L"',
     + 1X,5('*')/5X,'*',61X,'*'/5X,63('*')/
     + //30X,12('*')/30X,'RUN NO. =',I3/30X,12('*')///)
311   FORMAT(//5X,63('*')/5X,'*',61X,'*'/6('*'),1X,
```

```
     +'OUTPUT FILE OF THE IN-SITU COMBUSTION SIMULATOR :"OUTPUT.S"',
     + 1X,5('*')/5X,'*',61X,'*'/5X,63('*')/
     + //30X,12('*')/30X,'RUN NO. =',I3/30X,12('*')///)
315   FORMAT(27('*'),2X,'RESERVOIR DATA',2X,27('*')//
     +'Reservoir Length (x-direction)',8X,'=',F9.3,2X,'ft'/
     +'Reservoir Width  (y-direction)',8X,'=',F9.3,2X,'ft'/
     +'Reservoir Height (z-direction)',8X,'=',F9.3,2X,'ft'/
     +'Number of Grid Blocks in x-direction ',1X,'=',I9/
     +'Number of Grid Blocks in y-direction ',1X,'=',I9/
     +'Number of Grid Blocks in z-direction ',1X,'=',I9/
     +'Length of Each Block in x-direction ',2X,'=',F9.3,2X,'ft'/
     +'Length of Each Block in y-direction ',2X,'=',F9.3,2X,'ft'/
     +'Length of Each Block in z-direction ',2X,'=',F9.3,2X,'ft'/
     +'Slope in x-direction',18X,'=',F9.2,2X,'degrees'/
     +'Slope in y-direction',18X,'=',F9.2,2X,'degrees'//
     +'Permeability',2X,'=',F6.2,2X,'Darcy'/
     +'Porosity',6X,'=',F6.2//)
320   FORMAT(/27('*'),2X,'INJECTION DATA',2X,27('*')//
     +'Injection Rates (lb-mol/d)'/27('-')/
     +'Water',5X,'=',F8.2/
     +'Heavy Oil',1X,'=',F8.2/
     +'Light Oil',1X,'=',F8.2/
     +'Inert Gas',1X,'=',F8.2/
     +'Oxygen',4X,'=',F8.2//
     +'Injection Temperature',1X,'=',F7.2,1X,'F',///)
321   FORMAT(26('*'),2X,'BAND HEATER DATA',2X,26('*'))
322   FORMAT(/'Band Heater Constant   =',F8.3,2X,'Btu/cu.ft.d (F)'/
     +'Preheat Time           =',F8.3,2X,'days')
323   FORMAT(//26X,'Band Heater Heat (F)',/72('-')/5(F11.5,3X))
325   FORMAT(///23('*'),2X,'NUMERICAL CONTROL DATA',2X,23('*')//
     +'Initial Timestep',15X,'=',F8.3,2X,'days'/
     +'Maximum Timestep',15X,'=',F8.3,2X,'days'/
     +'Stopping End Time',14X,'=',F8.3,2X,'days'/
     +'Output Frequency (day)',9X,'=',F8.3,2X,'days'/
     +'Output Frequency (timestep)',4X,'=',I8/
     +'Maximum Number of Timestep',5X,'=',I8/
     +'Max. Newton Iter. Per Timestep',1X,'=',I8///
     +'Timestep Change Norms',24X,'Convergence Tolerances'/
     + 26('-'),19X,24('-')/
     +'Pressure',5X,'=',F7.2,2X,'psi',19X,'Pressure',5X,'=',F9.5/
     +'Water Satur.',1X,'=',F7.2,2X,22X,'Water Satur.',1X,'=',F9.5/
     +'Oil Satur.',3X,'=',F7.2,2X,22X,'Oil Satur.',3X,'=',F9.5/
     +'Temperature',2X,'=',F7.2,2X,'F',21X,'Temperature',2X,'=',F9.5/
     +'Oxygen',7X,'=',F7.2,2X,22X,'Oxygen',7X,'=',F9.5/
     +'Heavy Oil',4X,'=',F7.2,2X,22X,'Heavy Oil',4X,'=',F9.5)
```

```
327   FORMAT(//'Data for Solving Linear System'/30('-'))
801   FORMAT('Solution Method = LU Decomposition')
820   FORMAT('Preconditioning = --'/'Scaling         = --'/
     +'Max. No.of Iter.= --'/'Tolerance       = --'//)
802   FORMAT('Solution Method = GMRES(',I2,')')
803   FORMAT('Solution Method = ORTHOMIN(',I2,')')
804   FORMAT('Solution Method = BI-CGSTAB')
805   FORMAT('Preconditioning = Yes')
806   FORMAT('Preconditioning = No')
807   FORMAT('Scaling         = Yes')
808   FORMAT('Scaling         = No')
809   FORMAT('Max. No.of Iter.= ',I3/ 'Tolerance       = ',D7.1//)
330   FORMAT(/25('*'),2X,'INITIAL CONDITIONS',2X,25('*'))
335   FORMAT(///18X,36('*'),/
     +19X,'The Time',8X,'=',F11.6,2X,'days'/
     +19X,'Timestep Size',3X,'=',F11.6,2X,'days'/
     +19X,'Timestep No.',4X,'=',I11,/18X,36('*'),///)
500   FORMAT(//26X,'Gas Pressure (psi)',/72('-')/5(F11.5,3X))
501   FORMAT(//26X,'Oil Pressure (psi)',/72('-')/5(F11.5,3X))
502   FORMAT(//25X,'Water Pressure (psi)',/72('-')/5(F11.5,3X))
503   FORMAT(//28X,'Gas Saturation',/72('-')/5(F11.7,3X))
504   FORMAT(//28X,'Oil Saturation',/72('-')/5(F11.7,3X))
505   FORMAT(//26X,'Water Saturation',/72('-')/5(F11.7,3X))
506   FORMAT(//28X,'Temperature (F)',/72('-')/5(F11.5,3X))
507   FORMAT(//16X,'Y1 - Water Mole Fraction in Gas Phase',/
     +72('-')/5(F11.7,3X))
508   FORMAT(//14X,'Y2 - Heavy Oil Mole Fraction in Gas Phase',/
     +72('-')/5(F11.7,3X))
509   FORMAT(//14X,'Y3 - Light Oil Mole Fraction in Gas Phase',/
     +72('-')/5(F11.7,3X))
510   FORMAT(//14X,'Y4 - Inert Gas Mole Fraction in Gas Phase',/
     +72('-')/5(F11.7,3X))
511   FORMAT(//16X,'Y5 - Oxygen Mole Fraction in Gas Phase',/
     +72('-')/5(F11.7,3X))
512   FORMAT(//13X,'X2 - Heavy Oil Mole Fraction in Liquid Phase',/
     +72('-')/5(F11.7,3X))
513   FORMAT(//13X,'X3 - Light Oil Mole Fraction in Liquid Phase',/
     +72('-')/5(F11.7,3X))
514   FORMAT(//18X,'Coke Concentration (lb-mol/ft3.d)',/
     +72('-')/5(F11.7,3X))
515   FORMAT(//24X,'Gas Density (lb-mol/ft3)',/72('-')/5(F11.5,3X))
516   FORMAT(//24X,'Oil Density (lb-mol/ft3)',/72('-')/5(F11.5,3X))
517   FORMAT(//23X,'Water Density (lb-mol/ft3)',/72('-')/5(F11.5,3X))
518   FORMAT(//26X,'Gas Viscosity (cp)',/72('-')/5(F11.5,3X))
519   FORMAT(//26X,'Oil Viscosity (cp)',/72('-')/5(F11.5,3X))
```

```
520   FORMAT(//26X,'Water Viscosity (cp)',/72('-')/5(F11.5,3X))
521   FORMAT(//24X,'Ekilibrium Constant - K1',/72('-')/5(F11.5,3X))
522   FORMAT(//24X,'Ekilibrium Constant - K2',/72('-')/5(F11.5,3X))
523   FORMAT(//24X,'Ekilibrium Constant - K3',/72('-')/5(F11.5,3X))
524   FORMAT(//23X,'Gas Relative Permeability',/72('-')/5(F11.5,3X))
525   FORMAT(//23X,'Oil Relative Permeability',/72('-')/5(F11.5,3X))
526   FORMAT(//22X,'Water Relative Permeability',/72('-')/5(F11.5,3X))
527   FORMAT(//16X,'Reaction Rate - rA - (lb-mol/ft3.d)',/
     +72('-')/5(E11.4,3X))
528   FORMAT(//16X,'Reaction Rate - rB - (lb-mol/ft3.d)',/
     +72('-')/5(E11.4,3X))
529   FORMAT(//16X,'Reaction Rate - rC - (lb-mol/ft3.d)',/
     +72('-')/5(E11.4,3X))
530   FORMAT(//16X,'Reaction Rate - rD - (lb-mol/ft3.d)',/
     +72('-')/5(E11.4,3X))
531   FORMAT(//19X,'Gas Internal Energy (Btu/lb-mol)',/
     +72('-')/5(E11.4,3X))
532   FORMAT(//19X,'Oil Internal Energy (Btu/lb-mol)',/
     +72('-')/5(E11.4,3X))
533   FORMAT(//18X,'Water Internal Energy (Btu/lb-mol)',/
     +72('-')/5(E11.4,3X))
534   FORMAT(//18X,'Rock Internal Energy (Btu/lb-mol)',/
     +72('-')/5(E11.4,3X))
535   FORMAT(//23X,'Coke Enthalpy (Btu/lb-mol)',/
     +72('-')/5(E11.4,3X))
375   FORMAT(///20('*'),2X,'PRODUCTION RATES (lb-mol/d)',2X,21('*')//
     +'Component',5X,'Water Phase',5X,'Oil Phase',6X,'Gas Phase',10X,
     +'TOTAL',/9('-'),4X,12('-'),4X,11('-'),4X,11('-'),6X,11('-'),//
     +'Water',7X,F12.6,8X,4('-'),8X,F11.5,6X,F11.5,/
     +'Heavy Oil',8X,4('-'),7X,F11.5,5X,F11.5,6X,F11.5,/
     +'Light Oil',8X,4('-'),7X,F11.5,5X,F11.5,6X,F11.5,/
     +'Inert Gas',8X,4('-'),11X,4('-'),8X,F11.5,6X,F11.5,/
     +'Oxygen',11X,4('-'),11X,4('-'),8X,F11.5,6X,F11.5,///)
380   FORMAT(72('='))
C
      RETURN
      END
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
C
C**** DECLARATION
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER NN,MM,IT,IHGMR
      PARAMETER (NN=12,MM=864,IT=500,IHGMR=373680)
```

```
      INTEGER N,IHLS,IOUTFQ,ITNEWT,MSTEP,MELIMF,NGBX,NGBY,ITSTEP,MXIT
      INTEGER IFLCF1,IFLCF2,IRUNNO,MAXITE,MGMRES,ISCAL,ILUF,ITMET
      INTEGER ICITE,IFLCOK,IOUT3,NBLLIN
C
      COMMON / DAT  / IRUNNO
      COMMON / DAT1 / RLEN,RWID,DZ,NGBX,NGBY,SLOPEX,SLOPEY,G
      COMMON / DAT2 / PRESG(NN,NN),SW(NN,NN),SO(NN,NN),TEMP(NN,NN),
     *Y5(NN,NN),X2(NN,NN),CC(NN,NN),BHTEMP(NN,NN)
      COMMON / DAT3 / PRHT,BTHCD,PERM,THCD,THCDCAP,GASCST,PORE,TREF,PREF
      COMMON / DAT4 / PINDEX,PRODP,COKMAX,IHLS
      COMMON / DAT5 / QINJ1(NN,NN),QINJ2(NN,NN),QINJ3(NN,NN),
     *QINJ4(NN,NN),QINJ5(NN,NN),TINJ
      COMMON / DAT6 / CPG1,CPG2,CPG3,CPG4
      COMMON / DAT7 / EQUW2,EQUW1,EQUW3,EQUHO1,EQUHO2,EQUHO3,EQULO1,
     *EQULO3
      COMMON / DAT8 / DW1,DW2,DW3
      COMMON / DAT9 / DHO1,DHO2,DHO3,DLO1,DLO2,DLO3
      COMMON / DAT10/ VG15,VG14,VG13,VG12,VG11
      COMMON / DAT11/ VG25,VG24,VG23,VG22,VG21
      COMMON / DAT12/ VW1,VW2,VW3,VW4
      COMMON / DAT13/ VLO1,VLO2,VHO1,VHO2
      COMMON / DAT14/ WTML1,WTML2,WTML3,WTML4,WTML5,WTML6
      COMMON / DAT15/ AKRWRO,AKRGRO,AKROCW,SWC,SORW,SORG,SGC,ZW,ZG,ZOG,
     *ZOW
      COMMON / DAT16/ CWA,CHO,CLO,CIG,CO2
      COMMON / DAT17/ EHO1,TCLO,TCHO,EW1,TCWT
      COMMON / DAT18/ CCK,CRK,ARHA,ARHB,ARHC,ARHD
      COMMON / DAT19/ EENA,EENB,EENC,EEND,HRA,HRB,HRC,HRD
      COMMON / DAT20/ S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12
      COMMON / DAT21/ DTINIT,DTMAX,ETIM,URMX,MXIT,MSTEP,IOUTFQ
      COMMON / DAT88/ OUTFQD
      COMMON / DAT22/ PNORM,SWNORM,SONORM,TNORM,YNORM,XNORM
      COMMON / DAT23/ PCON,SWCON,SOCON,TCON,YCON,XCON
      COMMON / DAT24/ ITMET,MGMRES,ILUF,ISCAL,MAXITE,IOUT3
      COMMON / DAT25/ TOL
      COMMON / INT1 / N,ITNEWT,MELIMF,ITSTEP
      COMMON / INT2 / IFLCF1,IFLCF2,ICITE,IFLCOK,NBLLIN
      COMMON / DOO1 / DX,DDDX,DY,DDDY,DTOLD,DT
      COMMON / EOO1 / EALPR,EALW,EALO,EALEN,EALY5,EALX2,
     *EQW,EQHO,EQLO,EQIG,EQO2,EQEB
      COMMON / QOO1 / QHOLD1,QHOLD2,QHOLD3,QHOLD4,QHOLD5
      COMMON / TOO1 / TIME
      COMMON / VOO1 / VOL
      COMMON / AOO1 / AKRG(NN,NN),AKRO(NN,NN),AKRW(NN,NN)
      COMMON / DOO1 / DENG(NN,NN),DENO(NN,NN),DENW(NN,NN)
```

```
      COMMON / E001 / EK1(NN,NN),EK2(NN,NN),EK3(NN,NN)
      COMMON / H001 / HG(NN,NN),HW(NN,NN),HO(NN,NN),HC(NN,NN)
      COMMON / P001 / PRESO(NN,NN),PRESW(NN,NN),PTTEMP(NN,NN),
     *PTPRESG(NN,NN),PTSG(NN,NN),PTSO(NN,NN),PTSW(NN,NN),PTX2(NN,NN),
     *PTX3(NN,NN),PTY1(NN,NN),PTY2(NN,NN),PTY3(NN,NN),PTY4(NN,NN),
     *PTY5(NN,NN),PTDENG(NN,NN),PTDENO(NN,NN),PTDENW(NN,NN),
     *PTUG(NN,NN),PTUO(NN,NN),PTUW(NN,NN),PTUR(NN,NN),PTCC(NN,NN),
     *PTHC(NN,NN)
      COMMON / Q001 / QPRO1(NN,NN),QPRO2(NN,NN),QPRO3(NN,NN),
     *QPRO4(NN,NN),QPRO5(NN,NN),QP1G(NN,NN),QP2G(NN,NN),QP3G(NN,NN)
      COMMON / R001 / RA(NN,NN),RB(NN,NN),RC(NN,NN),RD(NN,NN)
      COMMON / S001 / SG(NN,NN)
      COMMON / U001 / UG(NN,NN),UO(NN,NN),UW(NN,NN),UR(NN,NN)
      COMMON / V001 / VISG(NN,NN),VISO(NN,NN),VISW(NN,NN)
      COMMON / W001 / WTMLG(NN,NN),WTMLO(NN,NN),WTMLW(NN,NN)
      COMMON / X001 / X3(NN,NN)
      COMMON / Y001 / Y1(NN,NN),Y2(NN,NN),Y3(NN,NN),Y4(NN,NN)
      COMMON / HLOS / TINIT(NN,NN),HLOSS(NN,NN),PHL(NN,NN),QHL(NN,NN),
     *ENCAP(NN,NN),PTPHL(NN,NN),PTQHL(NN,NN),PTENCAP(NN,NN),PTDHL,DHL
      COMMON / MAT1 / A(MM,MM),B(MM),DEL(MM),XO(MM),W(MM,MM),U(MM,MM)
      COMMON / MAT2 / EM(MM),SOL(MM),RHST(MM),BD(MM),R(IHGMR),V(MM,MM)
      COMMON / MAT3 / H(IHGMR),GE(MM),RO(MM),PO(MM),VO(MM),R1(MM)
      COMMON / MAT4 / P1(MM),V1(MM),X1(MM),S(MM),T(MM),Q(MM),Z(MM)
      COMMON / MAT5 / P(MM,IT),AP(MM,IT),BETAN(IT)
C
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Appendix D

An Example Input Data File of The Simulator

```
  ********************************************************************
  *                                                                  *
*** INPUT DATA FILE OF THE IN-SITU COMBUSTION SIMULATOR :"INPUT.DAT" ***
  *                                                                  *
  ********************************************************************


                                RUN NO.
                                  10


**************************  RESERVOIR DATA  **************************

    Reservoir Dimensions (ft)                Number of Grid Blocks
---------------------------------          --------------------------
Length          Width         Depth          x-direction     y-direction
164.0           115.0          21.0              10               1


        Slopes(degrees)
------------------------------
x-direction       y-direction                   Gravity(ft/s**2)
    0.0               0.0                            32.18
```

********** INITIAL CONDITIONS **********

Initial Reservoir Pressure(psi)

65.95 65.95 65.95 65.95 65.95 65.95 65.95 65.95 65.95 65.95

Initial Water Saturation

0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2

Initial Oil Saturation

0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5

Initial Temperature(F)

200.0 200.0 200.0 200.0 200.0 200.0 200.0 200.0 200.0 200.0

Y5 - Initial Oxygen Mole Fraction in Gas Phase

0.00001 0.00001 0.00001 0.00001 0.00001 0.00001 0.00001 0.00001 0.00001
0.00001

X2 - Initial Heavy Oil Mole Fraction in liquid Phase

0.99999 0.99999 0.99999 0.99999 0.99999 0.99999 0.99999 0.99999 0.99999
0.99999

Initial Coke Concentration (lb-mol/cu.ft)

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

********** BAND HEATER DATA **********

Band Heater Heat (F)

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

Preheat Time(d)	Band Heater Const.(Btu/cu.ft.d(F))
0.0	0.0

********** SOME CONSTANTS **********

Perm. (Darcy)	Ther.Cond.(sand) (Btu/ft.d.(F))	Ther.Cond.(cap rock) (Btu/ft.d.(F))	Gas Constant (Btu/lb-mol.(R))
4.0	38.4	38.4	1.9872

Porosity	Tref(R)	Pref(psi)	Prod.Const.	Prod.Pres.(psi)
0.38	536.4	65.23	1.0	60.0

Max.Coke Concent. (lb-mol/cu.ft)	Heat Loss (0=off, 1=on)
0.07692	1

********** INJECTION DATA **********

Injection Rates(lb-mol/d)				
Water	Heavy Oil	Light Oil	Inert Gas	Oxygen
0.0	0.0	0.0	0.0	300.0

Injec.Temp(F)
200.0

********** CAPILLARY PRESSURES **********

ACPG	BCGP	ACPW	BCPW
0.0	0.0	0.0	0.0

********** EQUILIBRIUM K-VALUES DATA **********

K1

BK1	AK1	CK1
116.0	459.69	4.464

K2

AK2	BK2	CK2
12.12767	-6738.91	167.13

K3

AK3	BK3	CK3
11.7803	-3370.43	45.29

********** DENSITIES DATA (lb-mol/cu.ft) **********

Water

AWD	BWD	CWD
3.464635	3.102641D-05	1.166667D-04

Oil

COD2	BOD2	AOD2	COD3	BOD3	AOD3
3.82D-04	1.0D-05	3.995	7.69D-04	2.2D-04	2.2

********** VISCOSITIES DATA (cp) **********

Gas

AGV5	AGV4	AGV3	AGV2	AGV1
2.1960D-04	2.1267D-04	0.2166D-04	0.03926D-04	0.08822D-04

BGV5	BGV4	BGV3	BGV2	BGV1
0.721	0.702	0.943	1.102	1.116

Water

AWV	BWV	CWV	DWV
1.0	0.14	0.03333	9.0D-06

Oil

AOV3	BOV3	AOV2	BOV2
0.02083	959.59	3.62431D-04	8485.4

********** MOLECULAR WEIGHT OF COMPONENTS (lb-mol/lb) **********

Water	Heavy Oil	Light Oil	Inert Gas	Oxygen	Coke
18.0	170.0	44.0	44.0	32.0	13.0

********** RELATIVE PERMEABILITIES DATA **********

Krwro	Krgro	Krocw	Swc	Sorw	Sorg	Sgc	Zw	Zg	Zog	Zow
0.25	0.7	1.0	0.2	0.3	0.09	0.05	3.0	1.0	3.0	3.0

********** ENTHALPIES DATA (Btu/lb-mol) **********

Gas

GE1	GE2	GE3	GE4	GE5
8.001895733	57.80473933	115.1317535	10.98767772	7.691374406

Oil			Water	
OEV1	Tc3(R)	Tc2(R)	WEV1	Tcw(R)
1918.895	665.6	1184.9	1655.4768	1164.78

Coke	Rock
CIE	RIE

4.060663506 35.0

********** REACTION KINETIC DATA **********

Arrhenius Constants (1/d.psi ,only AC 1/d)

AA	AB	AC	AD
1.0D+06	1.0D+06	3.0D+05	1.0D+06

Activation Energy (Btu/lb-mol)

EA	EB	EC	ED
33300.0	33300.0	28800.0	23400.0

Heat of Reaction (Btu/lb-mol)

HA	HB	HC	HD
948000.0	3490000.0	20000.0	225000.0

Stoichiometric Coefficients

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
5.0	3.0	4.0	18.0	12.0	13.0	2.0	4.67	1.33	1.25	1.00	0.5

********** NUMERICAL CONTROL DATA **********

Initial Time Step(d)	Max.Time Step(d)	End Time(d)
0.07	2.0	200.0

Urmax	Max.No.of Newton's It.	Max.No.of Step
0.5	50	900

Output Freq.(Time Step)	Output Freq.(Day)
50	30.0

Time Step Change Norms

Pres.(psi)	Water Sat.	Oil Sat.	Temper.(R)	Oxygen	Heavy Oil
57.98	0.2	0.2	72.0	0.2	0.2

Convergence Tolerances

Pressure	Water Sat.	Oil Sat.	Temperature	Oxygen	Heavy Oil
0.001	0.0001	0.0001	0.001	0.0001	0.0001

Data for Solving Linear System

Solution Method (see KEY)	m (for GMRES and ORTHOMIN)	Preconditioning (Yes(1)/No(0))
2	5	1

Scaling (Yes(1)/No(0))	Max.No.of iter. MAXIT	Tolerance (||r||<TOL)
1	300	0.1D-10

********** OUTPUT FILE **********

Do you want a long output file ? (Yes(1)/No(0))
0

****
*
********** THE END OF THE INPUT FILE **********
*
****

KEY :

LU Decomposition	: 1
GMRES	: 2
ORTHOMIN	: 3
IB-CGSTAB	: 4

Appendix E

Example Output Files of The Simulator

E.1 Numerical Output File

```
*********1*********2*********3*****MM**4*********5*********6*********7**

     ****************************************************************
     *                                                              *
****** OUTPUT FILE OF THE IN-SITU COMBUSTION SIMULATOR :"OUTPUT.N" *****
     *                                                              *
     ****************************************************************


                              ************
                              RUN NO. = 16
                              ************


***************************  RESERVOIR DATA  ****************************

Reservoir Length (x-direction)        =  164.000  ft
Reservoir Width  (y-direction)        =  115.000  ft
Reservoir Height (z-direction)        =   21.000  ft
Number of Grid Blocks in x-direction  =        4
Number of Grid Blocks in y-direction  =        4
Number of Grid Blocks in z-direction  =        1


***********************  NUMERICAL CONTROL DATA  ***********************
```

```
Initial Timestep               =   0.050  days
Maximum Timestep               =   1.000  days
Stopping End Time              =  10.000  days
Output Frequency (day)         =  20.000  days
Output Frequency (timestep)    =      10
Maximum Number of Timestep     =       5
Max. Newton Iter. Per Timestep =      50


Timestep Change Norms                    Convergence Tolerances
-------------------------                ----------------------
Pressure     =  57.98  psi               Pressure     =  0.00100
Water Satur. =   0.20                    Water Satur. =  0.00010
Oil Satur.   =   0.20                    Oil Satur.   =  0.00010
Temperature  =  72.00  F                 Temperature  =  0.00100
Oxygen       =   0.20                    Oxygen       =  0.00010
Heavy Oil    =   0.20                    Heavy Oil    =  0.00010


Data for Solving Linear System
------------------------------
Solution Method = GMRES( 5)
Preconditioning = Yes
Scaling         = Yes
Max. No.of Iter.= 300
Tolerance       = 0.1D-10


KEY
---------------------------------------------------------------------------
TSN  : Time Step No.
TSS  : Time Step Size
RRT  : Real Reservoir Time
NIPTS: Newton Iteration Per Time Step
LIPTS: Linear Iteration Per Time Step
TLI  : Total Linear Iteration
TNI  : Total Newton Iteration
FFL1 : Fail Flag 1 (Max number of Newton Iteration exceeded)
FFL2 : Fail Flag 2 (Results from the Newton Iteration are unacceptable)

***************************************************************************
```

```
TSN       TSS          RRT      NIPTS  LIPTS   TLI      TNI    FFL1  FFL2
---   ----------   -------   -----  -----  ------  -----  ----  ----
 1    0.05000000    0.0500     17     77      77     17
 2    0.03722365    0.0872     16     62     139     33
 3    0.04170449    0.1289     17     69     208     50
 4    0.05334335    0.1823     17     75     283     67
 5    0.06878673    0.2511     17     79     362     84


                ****************************************
                STOPPING MAX. NUMBER OF TIMESTEP REACHED

                    THE TIME         =   0.251058  days
                    TIMESTEP NO.     =          5
                ****************************************

                ****************************************
                *                                      *
*****************       THE END OF THE OUTPUT FILE     *****************
                *                                      *
                ****************************************
```

E.2 Short Output File

```
*********1*********2*********3*****MM**4*********5*********6*********7**


     ************************************************************
     *                                                          *
****** OUTPUT FILE OF THE IN-SITU COMBUSTION SIMULATOR :"OUTPUT.S" *****
     *                                                          *
     ************************************************************


                            ************
                            RUN NO. = 10
                            ************
```

```
*************************  RESERVOIR DATA  *************************

Reservoir Length (x-direction)        =  164.000  ft
Reservoir Width  (y-direction)        =  115.000  ft
Reservoir Height (z-direction)        =   21.000  ft
Number of Grid Blocks in x-direction  =       10
Number of Grid Blocks in y-direction  =        1
Number of Grid Blocks in z-direction  =        1
Length of Each Block in x-direction   =   16.400  ft
Length of Each Block in y-direction   =  115.000  ft
Length of Each Block in z-direction   =   21.000  ft
Slope in x-direction                  =     0.00  degrees
Slope in y-direction                  =     0.00  degrees

Permeability  =  4.00  Darcy
Porosity      =  0.38


*************************  INJECTION DATA  *************************

Injection Rates (lb-mol/d)
--------------------------
Water     =    0.00
Heavy Oil =    0.00
Light Oil =    0.00
Inert Gas =    0.00
Oxygen    =  300.00

Injection Temperature = 200.00 F


*************************  BAND HEATER DATA  *************************

Band Heater Constant   =   0.000  Btu/cu.ft.d (F)
Preheat Time           =   0.000  days


                        Band Heater Heat (F)
-----------------------------------------------------------------------
    0.00000       0.00000       0.00000       0.00000       0.00000
    0.00000       0.00000       0.00000       0.00000       0.00000
```

```
********************  NUMERICAL CONTROL DATA  ********************

Initial Timestep               =   0.050  days
Maximum Timestep               =   2.000  days
Stopping End Time              =  50.000  days
Output Frequency (day)         =  25.000  days
Output Frequency (timestep)    =     900
Maximum Number of Timestep     =     900
Max. Newton Iter. Per Timestep =      50


Timestep Change Norms                    Convergence Tolerances
-------------------------                ----------------------
Pressure     =  57.98  psi               Pressure     =  0.00100
Water Satur. =   0.20                    Water Satur. =  0.00010
Oil Satur.   =   0.20                    Oil Satur.   =  0.00010
Temperature  =  72.00  F                 Temperature  =  0.00100
Oxygen       =   0.20                    Oxygen       =  0.00010
Heavy Oil    =   0.20                    Heavy Oil    =  0.00010


Data for Solving Linear System
------------------------------
Solution Method = LU Decomposition
Preconditioning = --
Scaling         = --
Max. No.of Iter.= --
Tolerance       = --



*********************  INITIAL CONDITIONS  *********************

                  ***********************************
                   The Time        =   0.000000  days
                   Timestep Size   =   0.000000  days
                   Timestep No.    =           0
                  ***********************************
```

Gas Pressure (psi)

65.95000	65.95000	65.95000	65.95000	65.95000
65.95000	65.95000	65.95000	65.95000	65.95000

Oil Saturation

0.5000000	0.5000000	0.5000000	0.5000000	0.5000000
0.5000000	0.5000000	0.5000000	0.5000000	0.5000000

Water Saturation

0.2000000	0.2000000	0.2000000	0.2000000	0.2000000
0.2000000	0.2000000	0.2000000	0.2000000	0.2000000

Temperature (F)

200.00000	200.00000	200.00000	200.00000	200.00000
200.00000	200.00000	200.00000	200.00000	200.00000

Y5 - Oxygen Mole Fraction in Gas Phase

0.0000100	0.0000100	0.0000100	0.0000100	0.0000100
0.0000100	0.0000100	0.0000100	0.0000100	0.0000100

X2 - Heavy Oil Mole Fraction in Liquid Phase

0.9999900	0.9999900	0.9999900	0.9999900	0.9999900
0.9999900	0.9999900	0.9999900	0.9999900	0.9999900

Coke Concentration (lb-mol/ft3.d)

0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000	0.0000000

********** PRODUCTION RATES (lb-mol/d) **********

Component	Water Phase	Oil Phase	Gas Phase	TOTAL
Water	0.000000	----	0.00000	0.00000
Heavy Oil	----	0.00000	0.00000	0.00000
Light Oil	----	0.00000	0.00000	0.00000
Inert Gas	----	----	0.00000	0.00000
Oxygen	----	----	0.00000	0.00000

==========

****
The Time = 25.773792 days
Timestep Size = 2.000000 days
Timestep No. = 25
****

Gas Pressure (psi)

91.96812	91.08261	89.48698	87.03161	83.40101
78.54384	72.78665	68.89521	67.44135	66.46000

Oil Saturation

0.3919847	0.4310707	0.4614512	0.5002000	0.5309354
0.5575539	0.5680796	0.5335558	0.5073210	0.5010421

Water Saturation

0.0224841	0.1173881	0.1921682	0.2398521	0.2651883
0.2820566	0.2827580	0.2394612	0.2089874	0.2043832

Temperature (F)

302.93469	305.42580	305.33332	303.34473	299.33201
291.64277	271.23818	239.03558	222.63692	217.39177

Y5 - Oxygen Mole Fraction in Gas Phase

0.1495932	0.0690244	0.0382773	0.0222248	0.0136237
0.0100260	0.0117874	0.0156903	0.0158994	0.0147608

X2 - Heavy Oil Mole Fraction in Liquid Phase

0.9999975	0.9999961	0.9999947	0.9999931	0.9999912
0.9999877	0.9999773	0.9999639	0.9999594	0.9999589

Coke Concentration (lb-mol/ft3.day)

0.0000987	0.0002347	0.0004378	0.0007253	0.0009580
0.0009081	0.0005793	0.0003504	0.0002987	0.0002908

********** PRODUCTION RATES (lb-mol/d) **********

Component	Water Phase	Oil Phase	Gas Phase	TOTAL
Water	0.007799	----	113.09482	113.10262
Heavy Oil	----	2.06955	2.31650	4.38605
Light Oil	----	0.00008	36.93465	36.93474
Inert Gas	----	----	298.42057	298.42057
Oxygen	----	----	6.75337	6.75337

==========

****

STOPPING END TIME REACHED

```
   THE TIME        =  50.000000  days
   TIMESTEP NO.    =         64
***************************************

***********************************
 The Time        =  50.000000  days
 Timestep Size   =   0.313139  days
 Timestep No.    =         64
***********************************
```

Gas Pressure (psi)

130.68987	130.53144	130.34540	129.37188	125.63234
121.73985	117.35734	112.76372	108.07765	102.85546

Oil Saturation

0.0000000	0.0000000	0.4892742	0.6439270	0.5635253
0.5530673	0.5533851	0.5609575	0.5758878	0.5883821

Water Saturation

0.0000000	0.0000000	0.0000243	0.1832612	0.2594466
0.2829090	0.2938060	0.2935960	0.2916530	0.2951791

Temperature (F)

421.74841	541.94198	477.81832	296.01483	296.00880
292.26629	283.59812	273.85026	264.29826	250.54210

Y5 - Oxygen Mole Fraction in Gas Phase

1.0000000	0.9998554	0.0006869	0.0001102	0.0000198
0.0000040	0.0000011	0.0000004	0.0000002	0.0000001

X2 - Heavy Oil Mole Fraction in Liquid Phase

1.0000000	1.0000000	0.9998486	0.9998916	0.9998974
0.9998971	0.9998903	0.9998841	0.9998801	0.9998744

Coke Concentration (lb-mol/ft3.d)

0.0000000	0.0000000	0.0140953	0.0041642	0.0063599
0.0066486	0.0058019	0.0044429	0.0029555	0.0017960

********** PRODUCTION RATES (lb-mol/d) **********

Component	Water Phase	Oil Phase	Gas Phase	TOTAL
Water	603.662979	----	381.33387	984.99685
Heavy Oil	----	35.31849	9.27458	44.59306
Light Oil	----	0.00444	201.86643	201.87087
Inert Gas	----	----	672.66460	672.66460
Oxygen	----	----	0.00016	0.00016

==========

****
*
********** THE END OF THE OUTPUT FILE **********
*
****

Appendix F

Nomenclature

A_A	:	Arrhenius constant in Reaction A	$(psi.d)^{-1}$
A_B	:	Arrhenius constant in Reaction B	$(psi.d)^{-1}$
A_C	:	Arrhenius constant in Reaction C	d^{-1}
A_D	:	Arrhenius constant in Reaction D	$(psi.d)^{-1}$
C_{p_g}	:	Gas phase heat capacity	$lb-mol/ft^3.d$
C_{p_o}	:	Oil phase heat capacity	$lb-mol/ft^3.d$
C_c	:	Coke concentration	$lb-mol/ft^3.d$
$C_{c_{max}}$	:	Maximum coke concentration	$lb-mol/ft^3.d$
D	:	Depth	ft
E_A	:	Activation energy for Reaction A	$Btu/lb-mol$
E_B	:	Activation energy for Reaction B	$Btu/lb-mol$
E_C	:	Activation energy for Reaction C	$Btu/lb-mol$
E_D	:	Activation energy for Reaction D	$Btu/lb-mol$
$\mathbf{g}$	:	Gravity	ft/s^2
g_c	:	Gravity conversion constant	$ft.lb_m/s^2.lb_f$
h	:	Enthalpy	$Btu/lb-mol(^oR)$
h^*	:	Injection/production enthalpy	$Btu/lb-mol(^oR)$

H_A	:	Heat of Reaction A	$Btu/lb-mol$
H_B	:	Heat of Reaction B	$Btu/lb-mol$
H_C	:	Heat of Reaction C	$Btu/lb-mol$
H_D	:	Heat of Reaction D	$Btu/lb-mol$
H_{loss}	:	Heat loss to cap and base rock	$Btu/lb-mol$
H_{inj}	:	Heat injected by band heater	$Btu/lb-mol$
K	:	Permeability	$Darcy$
K_r	:	Relative permeability	
K_l	:	Phase equilibrium K-values (l=1,2,3)	
N_x	:	Number of gridblocks in x-direction	
N_y	:	Number of gridblocks in y-direction	
N_{gb}	:	$N_{gb} = N_x \times N_y$	
P	:	Pressure	psi
P_{cow}	:	Oil-water capillary pressure	psi
P_{cgo}	:	Oil-gas capillary pressure	psi
P_{prod}	:	Production pressure	psi
P_{ref}	:	Reference pressure	psi
q_l	:	Injection/production rate (l=g,o,w)	$lb-mol/d.ft^3$
q_l	:	Component injection/production rate	$lb-mol/d.ft^3$
		(l=1,2,3,4,5)	
r_A	:	Reaction rate of Reaction A	$lb-mol/ft^3.d$
r_B	:	Reaction rate of Reaction B	$lb-mol/ft^3.d$
r_C	:	Reaction rate of Reaction C	$lb-mol/ft^3.d$
r_D	:	Reaction rate of Reaction D	$lb-mol/ft^3.d$
R	:	Gas constant	$Btu/lb-mol(^oR)$
		(in gas density)	$psi.ft^3/lb-mol(^oR)$
s_l	:	Reactions stoichiometric coefficients	
		(l=1,2,...,12)	

S	:	Saturation	
S_{gc}	:	Critical gas saturation	
S_{wc}	:	Connate saturation	
S_{org}	:	Residual oil saturation (oil-gas system)	
S_{orw}	:	Residual oil saturation (oil-water system)	
t	:	Time	d
T	:	Temperature	^oR
T_B	:	Band heater temperature	^oR
T_{ref}	:	Reference temperature	^oR
U	:	Internal energy	$Btu/lb-mol$
V	:	Velocity	ft/d
X_2	:	Mole fraction of heavy oil in oil phase	
X_3	:	Mole fraction of light oil in oil phase	
x	:	Distance in x-direction	ft
Y_l	:	Mole fraction of component l in gas phase (l=1,2,3,4,5)	
y	:	Distance in y-direction	ft
z	:	Distance in z-direction	ft
γ	:	Productivity index constant	
η	:	Band heater constant	
Δx	:	Block length in x-direction	ft
Δy	:	Block length in y-direction	ft
Δz	:	Block length in z-direction	ft
Δt	:	Timestep	d
λ	:	Thermal conductivity	$Btu/d.ft(^oR)$
μ	:	Viscosity	cp
ρ	:	Density	$lb-mol/ft^3$
ϕ	:	Porosity	

Subscripts

c	:	Coke
g	:	Gas
o	:	Oil
r	:	Rock
w	:	Water
1	:	Water component
2	:	Heavy oil component
3	:	Light oil component
4	:	Inert gas component
5	:	Oxygen component
x, y	:	Directions in the Cartesian co-ordinate system
i	:	Grid block index in x-direction
j	:	Grid block index in y-direction

Superscripts

n	:	Old time level
$n+1$	:	New time level
Z_g	:	Power used in gas relative permeability relation
Z_w	:	Power used in water relative permeability relation
Z_{og}	:	Power used in oil-gas relative permeability relation
Z_{ow}	:	Power used in oil-water relative permeability relation

Appendix G

Glossary

API : American Petroleum Institute.

^oAPI : The specific gravity of oil is called ^oAPI and defined as:

$$^oAPI = \frac{141.5}{spgr60^o/60^o} - 131.5.$$

Barrel : A liquid volume measure equal to 42 *U.S.gallons.*

Centipoise (*cp*) **:** A unit of viscosity equal to 0.01 *poise.* A *poise* equals 1 $dyne - s.cm^2$.

Coke : A coherent, cellular, solid residue remaining from the dry distillation of oil.

Crude Oil : A mixture of hydrocarbons that exist in the liquid phase in natural underground reservoirs and remains liquid at atmospheric pressure after passing through processing facilities that separate out some components.

Darcy : A unit of permeability. A porous medium has a permeability of 1 *Darcy* when a pressure of 1 *atm* on a sample 1 ft long and 1 ft^2 in cross section will force a liquid of 1 *cp* viscosity through the sample at the rate of 1 ft^3/s.

Enthalpy : The sum of the internal energy of a system plus the product of the system's volume multiplied by the pressure exerted on the system by its surroundings.

Heat Capacity (Specific Heat) : The heat capacity is the quantity of energy required to raise the temperature of a unit amount of material by 1 degree.

Heavy Oil : Crude oil of $20^{o}API$ gravity or less.

Hydrostatic Pressure : The pressure at a point in a fluid at rest due to the weight of the fluid above it.

Inert Gas : A gas in group 0 of the periodic table of the elements; it is monatomic and, with limited exceptions, chemically inert.

Light Oil : Crude oil which has the gravity more than $20^{o}API$.

Miscible : Able to mix together; refers to two or more substances. Liquids that are not miscible separate into layers according to their specific gravities.

Oil(or Water) Saturation : The extend to which the voids rock contain, oil (or water) usually expressed in percent related to total void.

Permeability : Capacity of rock for transmitting a fluid. Degree of permeability depends upon the size and the shape of the pores and the size, shape and extend of the interconnections. The unit of permeability is the Darcy.

Polymer : A type of organic compound characterised by a large-chain molecule formed by thousands of repeating blocks called monomers; it is added to water for polymer flooding.

Pore Volume : The volume of void space in an oil reservoir which may contain petroleum, gas and/or brine.

Porosity : The fraction of the total volume of a material that is made up of empty space, or pore space. It is the volume of pore space expressed as a percentage of the total volume of the rock mass; measures the absorbent capacity of the material or the volume of the liquid held by the pores.

Reservoir : A discrete section of porous rock containing an accumulation of oil or gas, either separately or as a mixture.

Reservoir Fluids : Fluids contained within the reservoir under conditions of reservoir pressure.

Residual Oil: The amount of liquid petroleum remaining in the formation at the end of a specified production process.

Surfactant : A soluble compound that reduces the surface tension of liquids, or reduces interfacial tension between two liquids or a liquid and solid.

Thermal Conductivity : The theraml conductivity of a substance is a measure of the ability of that material to conduct energy.

Viscosity : The internal resistance offered by a fluid to flow. Attributable to the attraction between molecules of a liquid, this phenomenon is a measure of the combined effects of adhesion and cohesion between suspended particles and the liquid environment.

Volatile Component : A component of magma whose vapor pressure are high enough to allow them to be concentrated in any gaseous phase.

Appendix H

SI Metric Conversion Factors

Btu	x	1.055 056	E+ 00	=	kJ
Btu/lb_m	x	2.326	E+ 00	=	kJ/kg
cp	x	1.0	E− 03	=	$Pa.s$
$Darcy$	x	1.0	E+ 03	=	md
$Darcy$	x	1.062	E− 11	=	ft^2
ft	x	3.048	E− 01	=	m
ft^2	x	9.290 304	E− 02	=	m^2
ft^3	x	2.831 685	E− 02	=	m^3
ft/d	x	1.27	E+ 00	=	cm/hr
^oF		$(^oF - 32)/1.8$		=	^oC
^oF		$(^oF + 459.67)/1.8$		=	K
^oF		$^oF + 459.67$		=	^oR
lb_m	x	4.535 924	E− 01	=	kg
lb_m/ft^3	x	1.601 846	E+ 01	=	kg/m^3
psi	x	6.894 757	E+ 00	=	kPa
psi^{-1}	x	1.450 377	E+ 01	=	kPa^{-1}
psi/ft	x	2.262 059	E+ 01	=	kPa/m

Bibliography

[1] Aziz, K. and Settari, A., (1979), *Petroleum Reservoir Simulation*, Applied Science Publishers, London.

[2] Bailey, H.R. and Larkin, B.K., (1959), *Heat conduction in underground combustion*, Petroleum Trans. AIME, 216, pp.123-129.

[3] Baker, P.E., (1962), *Temperature profiles in underground combustion*, Soc. Pet. Eng. J., March, pp.21-27.

[4] Behie, A. and Vinsome, P.K.W., (1982) *Block iterative methods for fully implicit reservoir simulation*, paper SPE 9303 presented at the 55th Annual Fall Technical Conference, Dallas, September.

[5] Burden, R.L. and Faires, J.D., (1989), *Numerical Analysis*, PWS-KENT Publishing Company, USA.

[6] Burger, J.G., (1976), *Spontaneous ignition in oil reservoirs*, Soc. Pet. Eng. J., April, pp.73-81.

[7] Burger, J.G. and Sahuquet, B.C., (1972), *Laboratory research on wet combustion*, paper SPE 4144 presented at the 47th Annual Fall Meeting of SPE of AIME, San Antonio, Texas, October.

[8] Carcoana, A., (1990), *Results and difficulties of the World's largest in-situ combustion process: Suplacu de Barcau field, Romania*, paper SPE/DOE

20248 presented at the SPE/DOE 7th Symposium on Enhanced Oil Recovery, Tulsa, Oklahoma, April.

[9] Chandra, R., (1978), *Conjugate gradient methods for partial differential equations*, Ph.D. Thesis, Department of Computer Science, Yale University, New Haven, CT.

[10] Chu, C., (1963), *Two-dimensional analysis of radial heat wave*, J. Pet. Tech., October, pp.1137-1144.

[11] Chu, C., (1982), *State-of-the-art review of fireflood field projects*, J. Pet. Tech., January, pp.19-36.

[12] Coats, K.H., (1980), *In-situ combustion model*, Soc. Pet. Eng. J., December, pp.533-554.

[13] Cole, F.W., (1981), *Reservoir Engineering Manual*, Gulf Publishing Company, Houston, Texas.

[14] Concus, P. and Golub, G.H., (1976), *A generalized conjugate gradient method for nonsymmetric systems of linear equations*, in Lecture Notes in Economics and Mathematical Systems, 134, R.Glowinski and J.L. Lions, eds., Springer-Verlag, Berlin, pp.56-65.

[15] Crookston, R.B., Culham, W.E., and Chen, W.H., (1979), *A numerical simulation model for thermal recovery processes*, Soc. Pet. Eng. J., February, pp.37-58.

[16] Davies, R., (1988), *Mathematical modelling of in-situ combustion for enhanced oil recovery*, Ph.D. Thesis, The University of Bath.

[17] Donaldson, E.C., Chilingrian, G.V., and Yen, T.F., (1985), *Fundamentals and Analyses I, Developments in Petroleum Science*, 17A, Oxford, Elsevier.

[18] Eggenschwiler, M. and Farouq Ali, S.M., (1977), *Two-dimensional, single phase simulation of a fireflood*, Can-Venezuela Oil Sands, pp.487-495.

[19] Eisenstat, S.C., (1983), *A note on the generalized conjugate gradient method*, SIAM J. Numer. Anal., 20, pp.358-361.

[20] Eisenstat, S.C., Elman, H.C., and Schultz, M.H., (1983), *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20, pp.345-357.

[21] El-Khatib, N., (1973), *A mathematical model of thermal oil recovery in a forward in-situ combustion process*, Ph.D. Thesis, University of Pittsburgh.

[22] Elman, H.C., (1982), *Iterative methods for large, sparse, nonsymmetric systems of linear equations*, Ph.D. Thesis, Department of Computer Science, Yale University, New Haven, CT.

[23] Farouq Ali, S.M., (1977), *Multiphase, multidimensional simulation of in-situ combustion*, paper SPE 6896 presented at the 52nd Annual Fall Technical Conference and Exhibition of the SPE of AIME, Denver, October.

[24] Fletcher, R., (1975), *Conjugate gradient methods for indefinite systems*, in Proc. of the Dundee Biennial Conference on Numerical Analysis, G.A. Watson, ed., Springer-Verlag, New York, pp.73-89.

[25] Gerald, C.F., (1989), *Applied Numerical Analysis*, Addison-Wesley Publishing Company, USA.

[26] Golub, G.H. and Van Loan, C.F., (1989), *Matrix Computations*, The John Hopkins University Press, Baltimore and London.

[27] Gottfried, B.S., (1965), *A mathematical model of thermal recovery in linear systems*, Soc. Pet. Eng. J., September, pp.196-210.

[28] Grabowski, J.W., Vinsome, P.K., Lin, R.C., Behie, A., and Rubin, B., (1979), *A fully implicit general purpose finite difference thermal model for in-situ combustion and steam*, paper SPE 8396 presented at the SPE-AIME 54th Annual Fall Technical Conference and Exhibition of the SPE of AIME, Las Vegas, Nevada, September.

[29] Gregory, R.T. and Karney, D.L., (1969), *A Collection of Matrices for Testing Computational Algorithms*, Wiley, New York.

[30] Hageman, L.A. and Young, D.M., (1981), *Applied Iterative Methods*, Academic Press, New York.

[31] Hestenes, M.R. and Stiefel, E., (1952), *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49, pp.409-436.

[32] Javanmardi, G.R., (1992), *Forward in-situ combustion in fractured heavy oil reservoirs*, Ph.D. Thesis, The University of Bath.

[33] Johnson, C., (1990), *Numerical Solutions of Partial Differential Equations by The Finite Element Method*, Cambridge University Press, Cambridge, New York, New Rochelle, Melbourne, Sydney.

[34] Kershaw, D.S., (1978), *The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations*, J. Comput. Phys. 26, pp.43-65.

[35] Koniges, A.E. and Anderson, D.V., (1987), *ILUBCG2: a preconditioned bi-conjugate gradient routine for the solution of linear asymmetric matrix equations arising from 9-point discretizations*, Comput. Phys. Comm. 43, pp.297-302.

[36] Lanczos, C., (1952), *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Standards, 49, pp.33-53.

[37] Mikic, Z. and Morse, E.C., (1985), *The use of a preconditioned bi-conjugate gradient method for hybrid plasma stability analysis*, J. Comput. Phys., 61, pp.154-185.

[38] National Petroleum Council, (1984), *Enhanced Oil Recovery*, June.

[39] Oklany, J.S.F.A., (1992), *An in-situ combustion simulator for enhanced oil recovery*, Ph.D. Thesis, The University of Salford.

[40] Parrish, D.R. and Craig, F.F., Jr., (1969), *Laboratory study of a combination of forward combustion and waterflooding -the COFCAW process-*, J. Pet. Tech., June, pp.753-761.

[41] Peaceman, D.W., (1977), *Fundamentals of Numerical Reservoir Simulation*, Elsevier Scientific Publishing Co., Amsterdam, Oxford, New York.

[42] Prats, M., (1982), *Thermal Recovery*, SPE Monograph Vol 7, Dallas, Texas.

[43] Ramey, H. J., Jr., (1959), *Transient heat conduction during radial movement of a cylindrical heat source-applications to the thermal recovery process*, Petroleum Trans. AIME, 216, pp.115-122.

[44] Ramirez, W.F., (1987), *Application of Optimal Control Theory to Enhanced Oil Recovery*, Elsevier.

[45] Rubin, B. and Vinsome, P.K.W., (1980), *The simulation of the in-situ combustion process in one dimension using a highly implicit finite-difference scheme*, paper 79-30-14 presented at the 30th Annual Technical Meeting of the Pet. Soc. of CIM, Banff, Alberta, Canada.

[46] Saad, Y. and Schultz, M.H., (1986), *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7, pp.856-869.

[47] Smith, T., (1985), *Oil from setting the North Sea on fire*, New Scientist 105, pp.34-36.

[48] Smith, G.D., (1989), *Numerical Solution of Partial Differential Equations : Finite Difference Methods*, Third Edition, Oxford Applied Mathematics and Computing Science Series, Clarendon Press, Oxford.

[49] Smith, J.T. and Farouq Ali, S.M., (1971), *Simulation of in-situ combustion in a two-dimensional system*, paper SPE 3594 presented at the SPE-AIME 46th Annual Fall Meeting of SPE of AIME, New Orleans, October.

[50] Sonneveld, P., (1989), *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10, pp.36-52.

[51] Stiefel, E., (1955), *Relaxationmethoden bester strategie zur losung linearer glichungssystems*, Comm. Math. Helv., 29, pp.157-179.

[52] Thiez, P.L. and Lemonnier, P., (1988), *An in-situ combustion reservoir simulator with a new representation of chemical reactions*, paper SPE 17416 presented at the SPE California Regional Meeting, Long Beach, CA.

[53] Thomas, G.W., (1963), *A study of forward combustion in a radial system bounded by permeable media*, J. Pet. Tech., October, pp.1145-1149.

[54] Tuwil, A.A., (1991), *Horizontal producer wells in in-situ combustion (ISC) processes*, MPhil. Thesis, The University of Bath.

[55] Van der Vorst, H.A., (1981), *Iterative solution methods for certain sparse linear systems with a non-symmetric matrix arising from PDE-problems*, J. Comput. Phys. 44, pp.1-19.

[56] Van der Vorst, H.A., (1992), *BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13, pp.631-644.

[57] Varga, R.S., (1962) *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ.

[58] Vaughn, P., (1986), *A numerical model for thermal recovery process in tar sand: description and application*, Report No. DOE/FE/60177-2219.

[59] Vinsome, P.K.W., (1976), *Orthomin, an iterative method for solving sparse sets of simultaneous linear equations*, paper SPE 5729 in proceeding, Fourth Symposium on Reservoir Simulation, SPE of AIME, Los Angeles, February.

[60] Vinsome, P.K.W. and Westerveld, J., (1980), *A simple method for predicting cap and base rock heat losses in thermal reservoir simulators*, J. Can. Pet. Tech., July-September, pp.87-90.

[61] Widlund, O., (1978), *A Lanczos method for a class of nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 15, pp.801-812.

[62] Yortsos, Y.C. and Gavalas, G.R., (1982), *Heat transfer ahead of moving condensation fronts in thermal oil recovery processes*, Int. J. Heat. Mass Transfer, 25, pp.305-316.

[63] Young, D.M., (1971), *Iterative Solution of Large Linear Systems*, Academic Press, New York.

[64] Young, D.M. and Jea, K.C., (1980), *Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra and Appl., 34, pp.159-194.

[65] Youngren, G.K., (1980), *Development and application of an in-situ combustion reservoir simulator*, Soc. Pet. Eng. J., February, pp.39-51.